



MODELING AND SIMULATION  
SUPPORT FOR PARALLEL  
ALGORITHMS IN A HIGH-SPEED NETWORK

THESIS

Dustin E. Yates  
First Lieutenant, USAF

AFIT/GCS/ENG/97D

**DISTRIBUTION STATEMENT A**

Approved for public release;  
Distribution Unlimited

**DTIC QUALITY INSPECTED 3**

DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY

**AIR FORCE INSTITUTE OF TECHNOLOGY**

Wright-Patterson Air Force Base, Ohio

19980120 131

AFIT/GCS/ENG/97D-20

MODELING AND SIMULATION  
SUPPORT FOR PARALLEL  
ALGORITHMS IN A HIGH-SPEED NETWORK

THESIS

Dustin E. Yates  
First Lieutenant, USAF

AFIT/GCS/ENG/97D

Approved for public release; distribution unlimited

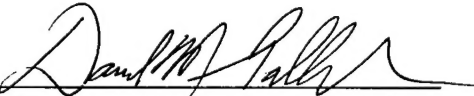
**DTIC QUALITY INSPECTED 3**


MODELING AND SIMULATION SUPPORT FOR  
PARALLEL ALGORITHMS IN A  
HIGH-SPEED NETWORK

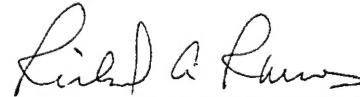
THESIS

Dustin E. Yates  
First Lieutenant, USAF

Presented to the Faculty of the Graduate School of  
Engineering of the Air Force Institute of Technology in  
Partial Fulfillment of the  
Requirements for the Degree of  
Master of Science in Computer Engineering

  
David M. Gallagher, Lt Col, USAF  
Member

  
Michael L. Talbert, Maj, USAF  
Member

  
Richard A. Raines, Maj, USAF  
Chairman

## *Acknowledgments*

Obviously there are many people to thank for their invaluable support that allowed me to complete this thesis. First, I'd like to thank my thesis advisor, Major Richard A. Raines, for his guidance and understanding during my battles with the formatting of this document. Also, I'd like to express my sincere appreciation to Lt. Col. David M. Gallagher and Major Michael L. Talbert for their support as members of my thesis committee. In addition, 1st Lt. David Gindhart needs to be recognized for "doing my work" and for providing the algorithms that fed the simulation model.

Furthermore, the ABA was instrumental in this work. Without this outlet to vent my frustrations, I might have not completed this thesis. Now we all can look forward to a life without the "Big Joe Knee."

Finally, I saved the most important for last. My family was so important during these past eighteen months that words can't express my deep appreciation and love for their support. Shannon, thanks for your incredible support and understanding throughout this journey. Girls, it's finally over!

Dustin E. Yates



## *Table of Contents*

Acknowledgments .....	iii
Table of Contents .....	iv
List of Figures .....	viii
Abstract .....	xii
I. Introduction.....	1
1.1 Background .....	1
1.2 Research Overview .....	2
1.3 Summary .....	3
II. Background.....	5
2.1 Introduction .....	5
2.1.1 Overview of NOW.....	5
2.1.1.1 First Generation Myrinet Switches.....	8
2.1.1.2 Second Generation Myrinet Switches .....	9
2.1.2 Discussion of NOW Research .....	10
2.1.3 Application of Past NOW Research to this Project.....	11
2.2 Ideology.....	11
2.2.1 Presentation of Message Layering.....	11
2.2.2 Perspective in Traffic Analysis.....	18
2.2.3 Overview of Past Traffic Modeling Research .....	21
2.2.4 Discussion of Myrinet Routing.....	22
2.3 Summary .....	33
2.3.1 Foundation of this Research Project.....	33
2.3.2 Divergence from Past NOW Research .....	34
III. Methodology .....	35

3.1 Introduction .....	35
3.1.1 Network Architectures.....	35
3.1.2 BONEs Designer .....	36
3.1.3 Traffic Analysis .....	37
3.1.4 Traffic Modeling.....	39
3.2 Design.....	40
3.2.1 Method.....	40
3.2.2 Assumptions .....	42
3.2.3 Construction.....	43
3.2.3.1 System Level Construction.....	43
3.2.3.2 Switch Level Construction .....	44
3.2.3.3 Workstation Level Construction.....	45
3.2.4 Parameters.....	47
3.2.4.1 Myrinet Parameters .....	47
3.2.4.2 Workstation Parameters.....	48
3.2.4.2.1 Common Workstation Parameters .....	49
3.2.4.2.2 Simulated Traffic Workstation Parameters .....	49
3.2.4.3 Simulation System Parameters .....	51
3.2.5 Measurements .....	53
3.2.6 Simulations .....	53
3.3 Verification and Validation.....	56
3.3.1 Design Verification.....	57
3.3.1.1 Construction .....	58
3.3.1.2 Measurements .....	61
3.3.1.3 Simulations.....	63

3.3.2 Model Validation .....	64
3.3.2.1 Parameters and Assumptions Validation .....	65
3.3.2.2 System Validation .....	65
3.3.2.3 Results Validation .....	66
3.4 Summary .....	67
IV. Analysis .....	69
4.1 Introduction .....	69
4.2 Traffic and Simulation Run Time Analysis .....	69
4.3 Trace Simulation Performance Analysis .....	73
4.4 Comparative Analysis .....	78
4.4.1 2-Node Topology Comparisons .....	78
4.4.2 4-Node Topology Comparisons .....	79
4.4.3 8-Node Topology Comparisons .....	80
4.4.4 16-Node Topology Comparisons .....	80
4.4.5 32-Node Topology Comparisons .....	81
4.4.6 64-Node Topology Comparisons .....	82
4.4.7 128-Node Topology Comparisons .....	83
4.4.8 Combined Topology Comparisons .....	83
4.4.9 Comparative Analysis Summary .....	86
4.5 Summary .....	86
V. Conclusion .....	91
5.1 Review of Thesis .....	91
5.2 Summary of Conclusions .....	92
5.3 Recommendations for Future Work .....	93
Appendix A: Statistical Analysis .....	95

A.1	Overview.....	95
A.2	Variance-Time Plot Analysis.....	95
A.3	Scaled Traffic Statistics.....	105
A.4	Statistical Analysis for Number of Independent Replications.....	110
Appendix B:	Designer Block Diagrams.....	113
B.1	Switch .....	113
B.2	Workstations.....	120
B.3	Simulation Topologies.....	125
B.4	Validation Tests.....	136
Appendix C:	Routing .....	138
C.1	Discussion.....	138
C.2	Examples .....	138
Appendix D:	Bibliography .....	144
Appendix E:	Vita .....	146

## *List of Figures*

Figure 2-1 Loopback Latency for Myricom's TCP/IP Stack.....	9
Figure 2-2 PM versus FM Latency [TeH96].....	15
Figure 2-3 Bandwidth of PM versus FM [TeH96].....	16
Figure 2-4 FM Latency Measurements for Sun Workstations and Pentium Pro Personal Computers [PaK97].....	17
Figure 2-5 FM Bandwidth Measurements for Sun Workstations and Pentium Pro Personal Computers [PaK97].....	18
Figure 2-6 Minimum Total Communication Size .....	21
Figure 2-7 3 x 3 Torus Network [Hu96] .....	30
Figure 2-8 Timeout and Throughput for Varying Worm Length [Hu96] .....	32
Figure 3-1 Myrinet Traffic Captures .....	38
Figure 3-2 Workload Targets in Bytes .....	51
Figure 3-3 Simulations .....	58
Figure 3-4 Theoretical Validation Test Number 1 Results .....	66
Figure 3-5 Theoretical Validation Test Number 2 Results .....	67
Figure 3-7 Validation Test Number 4 for Actual Measured Data vs. Simulated Data .....	68
Figure 3-6 Validation Test Number 3 for Actual Measured Data vs. Simulated Data .....	68
Figure 4-1 Row Column Basic Algorithm Trace Data .....	70
Figure 4-2 Packet Count v. Time for the Row Column Basic Algorithm for .0001 to .0075s .....	71
Figure 4-3 Packet Count v. Time for the Row Column Basic Algorithm for .01 to .075s Intervals .....	72
Figure 4-4 Simulation Execution Time in Calendar Days .....	74
Figure 4-5 Data Message Throughput for 2-Node Capture Simulations .....	75
Figure 4-6 Average Throughput at 90% Confidence Interval for 2-Node Simulations....	76

Figure 4-7 Data Message Throughput for the 4-Node Capture Simulations .....	76
Figure 4-8 2-Node Throughput for Stat Tests.....	79
Figure 4-9 4-Node Throughput for Stat Tests.....	79
Figure 4-10 8-Node Throughput for Stat Tests.....	80
Figure 4-11 16-Node Throughput for Stat Tests.....	81
Figure 4-12 32-Node Throughput for Stat Tests.....	82
Figure 4-13 64-Node Throughput for Stat Tests.....	83
Figure 4-15 Combined Throughput for Stat Tests .....	84
Figure 4-14 128-Node Throughput for Stat Tests.....	84
Figure 4-16 Combined Results for the Stat Tests Using Arithmetic (a) and Weighted (b & c) Means .....	85
Figure 4-17 Theoretical versus Achieved Bandwidth.....	88
Figure 4-18 Percentage Overhead Observed During Traffic Traces.....	89
Figure A-1 RC Basic 4-Node Variance-Time Plot for .0005 to .025 seconds.....	96
Figure A-2 RC Opt1 2-Node Variance-Time Plot for .00025 to .01 seconds.....	97
Figure A-3 RC Opt1 4-Node Variance-Time Plot .0005 to .025 seconds .....	98
Figure A-4 VR Basic 2-Node Variance-Time Plot .00025 to .01 seconds .....	99
Figure A-5 VR Basic 4-Node Variance-Time Plot .00025 to .01 seconds .....	100
Figure A-6 VR Opt2 2-Node Variance-Time Plot .00025 to .01 seconds .....	101
Figure A-7 VR Opt2 4-Node Variance-Time Plot .00025 to .01 seconds .....	102
Figure A-8 VR Opt4 2-Node Variance-Time Plot.00025 to .01 seconds .....	103
Figure A-9 VR Opt4 4-Node Variance-Time Plot .00025 to .01 seconds .....	104
Figure A-10 VR Basic Controller and Worker Statistics.....	105
Figure A-11 RC Opt1 Controller and Worker Statistics.....	106
Figure A-12 VR Basic Controller and Worker Statistics.....	107
Figure A-13 VR Opt2 Controller and Worker Statistics.....	108

Figure A-14 VR Opt4 Controller and Worker Statistics.....	109
Figure A-15 RC Basic Number of Independent Replications Statistics .....	110
Figure A-16 RC Opt1 Number of Independent Replications Statistics .....	111
Figure A-17 VR Basic Number of Independent Replications Statistics .....	111
Figure A-18 VR Opt2 Number of Independent Replications Statistics .....	112
Figure A-19 VR Opt4 Number of Independent Replications Statistics .....	112
Figure B-1 Myrinet 8-Port Switch Block Diagram.....	113
Figure B-2 Input Interpreter Block Diagram.....	114
Figure B-3 Myrinet Switch 8x4 Block Diagram.....	115
Figure B-4 Crossbar Switch Module 8 x 1 Block Diagram .....	116
Figure B-5 8-Port Multiplexor Block Diagram.....	117
Figure B-6 Cross Point Myrinet.....	118
Figure B-7 Arbitration 8-Port Block Diagram .....	119
Figure B-8 Pick Winner Block Diagram.....	120
Figure B-9 Myrinet Form Workstation Block Diagram.....	121
Figure B-10 Form Source Process Block Diagram.....	121
Figure B-11 Myrinet Adapter Block Diagram .....	122
Figure B-12 Generate Route Flit Block Diagram .....	123
Figure B-13 Myrinet Advanced (Stat) Workstation Block Diagram .....	123
Figure B-14 Advanced Source Process Block Diagram .....	124
Figure B-15 Advanced Destination Process Block Diagram .....	124
Figure B-16 2-Node Trace Topology Block Diagram .....	125
Figure B-17 4-Node Trace Topology Block Diagram .....	126
Figure B-18 2-Node Stat Topology Block Diagram .....	127
Figure B-19 4-Node Stat Topology Block Diagram .....	127

Figure B-20 8-Node Stat Topology Block Diagram .....	128
Figure B-21 16-Node Stat Topology Block Diagram .....	129
Figure B-22 Init/Wrap-Up Routines 144-Nodes Block Diagram .....	129
Figure B-23 16-Nodes (4 Switches) Block Diagram .....	130
Figure B-24 16-Nodes (3 Switches).....	131
Figure B-25 32-Node Stat Topology Block Diagram .....	131
Figure B-26 32-Nodes (9 Switches).....	132
Figure B-27 32-Node (6 Switches) Block Diagram.....	133
Figure B-28 64-Node Stat Topology Block Diagram .....	134
Figure B-29 128-Node Stat Topology Block Diagram .....	134
Figure B-30 64-Nodes (19 Switches) Block Diagram .....	135
Figure B-31 128-Nodes (39 Switches).....	136
Figure B-32 Two Node Validation Test Block Diagram .....	137
Figure B-33 Two-Node Validation Test Topology Number Two .....	137
Figure C-1 16-Workstations, 3-Switches Routing Table .....	141
Figure C-2 16-Workstations, 4-Switch Route Table for Configuration Number 1 .....	142
Figure C-3 16-Workstations, 4-Switches Route Table for Configuration Number 2 .....	143



## *Abstract*

This thesis investigates the ability of a simulation model to compare and contrast parallel processing algorithms in a high-speed network. The model extends existing modeling, analysis, and comparison of parallel algorithms by providing graphics based components that facilitate the measurement of system resources. Simulation components are based on the Myrinet local area network standard. The models provide seven different topologies to contrast the performance of five variations of Fast Fourier Transform (FFT) algorithms. Furthermore, the models were implemented using a commercially developed product that facilitates the testing of additional topologies and the investigation of hardware variations. Accurate comparisons are statistically validated and supported via common operating assumptions and the Myrinet standards. Based on the statistical confidence, the conclusion is drawn that a variation of a FFT algorithm based on row-column computations performs better than the other choices considered.

## ***I. Introduction***

### ***1.1 Background***

The need to solve computationally complex problems is best exemplified by the government's "Grand Challenges." The Grand Challenges, scientific and engineering problems once thought unsolvable, sparked a rush to build mammoth computers with application specific designs. Today's focus is not the same as it was in the early days of the search. Multi-millions of dollars were once a common expenditure for multiple government contracts. However, today we see few contracts and even fewer competitors for the government's dollars. It's long been the case that the lengthy development cycles and high cost expenditures were just part of the business. However, as technology has advanced, so have the ideas to effectively utilize the hardware. Thus, the government seeks to harness a combination of the best technology and the best algorithms.

Primarily resulting from the government's research dollars, the massively parallel class of multi-processors (MPPs) effectively increased the range of tractable problems. However, as the microprocessor came of age, most of the business thrust moved from the large computing platforms to the smaller, more economical workstations. When considering a workstation, the range of systems covers the stand-alone personal computer to the multi-user UNIX systems. Consequently, the development thrust toward these workstations provided the edge that the MPPs could not compete with, i.e., economies of scale. Anderson, et al, cover this phenomenon and note the lag time that MPPs incur when a system is fielded [AnC94]. Continuing this theme that commodity parts and availability drive the current technology market, Myricom fielded the low-cost, high-performance Myrinet [BoC95].

Myricom's Myrinet provides a high-speed local area network alternative for message passing based parallel processing.

Current research on parallel processing in high-speed networks focuses on improving the messaging layers that carry the end-to-end messages. Additionally, a limited amount of modeling and simulation targeted Myricom's high-speed network switch to incorporate proposed improvements for messaging layers that use such a component. However, that body of work failed to target specific applications for the Myrinet and potentially failed to accurately portray the traffic that is generated in a parallel processing environment. With the potential to maximize economies of scale, the Myrinet class of parallel processing networks needs to target specific applications to better increase its acceptance in to mainstream computing. Thus, as the next section overviews, this research will concentrate on developing a model to feed actual application generated data.

## ***1.2 Research Overview***

Currently, no research results have been found in published literature that evaluate the performance of parallel processing algorithms, specifically Fast Fourier Transforms (FFTs), in a Myrinet. This work concentrates on developing a scalable Myrinet model to simulate a set of FFTs to determine the performance and applicability of the model to simulate high-order topologies. The Myrinet platform was chosen due to its relatively low expense and due to its compatibility with available government computers. It is the second reason that appears to have the biggest impact on parallel computing. The fact that the Myrinet

conforms to many existing computing platforms places it in a position to take advantage of the millions of processing cycles that go idle every day.

Critical to the success of any model, the inputs for this study are given high priority. For this study, the network communications numbered well over one hundred thousand. The results obtained from analyzing these led to the use of a bursty traffic generator as opposed to a tradition Poisson model. The consequences of this decision are heavily investigated in more than thirty simulations as outlined in Chapter 3. Equally important to the inputs is an accurate reflection of the system under test. Chapter 2 outlines the technical specifics used for the model.

### ***1.3 Summary***

This chapter outlined the problem of interest- the applicability of a computer model to a Myrinet based parallel processing environment. The FFT was introduced as the algorithm of choice to generate network traffic across the Myrinet. In fact, as Chapters 2 and 3 will outline, five variations of the FFT are monitored to provide inputs to the model. Chapter 2 presents a discussion of the problems researched regarding the efficient use of networks such as Myrinet. In addition, the technical aspects of the Myrinet are introduced as a basis to build the model from.

Extending the information introduced in Chapter 2, Chapter 3 ties the traffic analysis and the technical specifics together to form the method employed to execute the simulations. In Chapter 4, the metric of interest, bandwidth is analyzed in relation to each of the algorithms tested, as well as, an analysis of the traffic patterns used as inputs for the model.

In Chapter 5, the conclusions drawn from this investigation are provided along with recommendations regarding future research.

## *II. Background*

### *2.1 Introduction*

The literature review contained here-in focuses on the use of Myricom's Myrinet Network Switch to implement a Network of Workstations (NOW). Discussions will focus on the applicability of a NOW environment to the parallel processing domain. Through modeling and simulation, this research effort seeks to exploit the network performance of the Myrinet focusing on parallel processing applications using the Fast Fourier Transform.

#### *2.1.1 Overview of NOW*

The definition of a NOW centers on a reliable network to provide a shared pool of computing capabilities. Proposed environments include UNIX workstations supported by a high-speed interconnection switch, advance personal computers/network servers interconnected by a high-speed switch, and traditional network infrastructures supporting the same systems. This research effort focuses on the first group through Sun's Sparc Ultra workstation and Myricom's Myrinet switch. However, the hardware is not the sole focus of the research. As discussed below in Section 2.2, the underlying software delivery components and the exploitation of the hardware through parallel processing actually comprise the areas of greatest research opportunities. Research in this subject area shows that hardware device technology exceeds the capability of the software [BoC95] [LaC97] [PaK97].

As stated above, Myricom's Myrinet switch interconnects the Sun workstations for this research effort. In [BoC95], Boden, et al, note that Myricom's switch provides two one-

way, high bandwidth links that provide full-duplex capability between two nodes. In addition, the network is considered very reliable with a bit error rate (BER) of approximately  $10^{-15}$  [BoC95]. The switch implements cut-through (wormhole) routing to fully interconnect the devices connected at each port. Operating at the physical layer of the OSI reference model, a Myrinet provides the capability for up to 2.56 Gbps full-duplex communications that support multi-protocol communications. Myrinet switches are further discussed in Subsections 2.1.1.1 and 2.1.1.2. Network topology within a Myrinet is considered arbitrary such that various models of interconnection networks are supported, such as torus, fat tree, and meshes, through multi-switch connections which allows scalability support through the multistage-switch connections [BoC95]. In [BoC95], the authors note that the use of this switch to mimic a classic massively parallel processor (MPP) message passing configuration actually results in less network overhead. To exemplify, they note that a single chip could replace the 16 mesh-routing chips used in the Intel Delta multi-computer resulting in fewer internal conflicts and reduced network diameter.

The interface to the Myrinet switch integrates the network and the Sun workstation via a custom VLSI chip called LANai. The interface supplies a region of fast static random access memory (SRAM) running via a programmable processor that allows standard operating system (OS) communications as well as customized messaging protocols. The interface's processor speed is surprisingly slow (37.5MHz) but performs adequately due to the fact that the Sun workstation's bus input/output bus (SBus) only operates at 25 MHz while supporting 123 Megabytes (MB) per second transfer bandwidth in peak performance

according to Sun<sup>1</sup>. Additionally, the adapter's processor is optimized to operate on each edge of the clock resulting in performance equal to a 75MHz clock, as well as, supporting modifications as it is a programmable processor. The chip also allows direct memory addressing (DMA) through support for memory transfers directly from system memory. The network interface along with the device driver accomplishes this by integrating the SRAM directly into the system memory space, which reduces the overhead work needed to transmit over the network [Boc95] [PaC97].

At the system level, the Sun workstations provide superior processing capabilities. The Ultra Sparc workstation provides high speed processing, 167 MHz clock rate with a SPECint rating of 250 and a SPECfp rating of 350, and ample dynamic random access memory (DRAM), 128 MB as configured in this research with support for up to 1 GB, for processing large parallel computations. The internal structure of the workstation's SBus introduces problems when attempting gigabit speed networking. Overcoming the SBus's relatively slow speed requires replacing the OS's traditional view of an unreliable network. Many research efforts show that the overhead in the messaging layers results in unneeded processing that can be eliminated [BoC95] [CIJ89] [LaC97] [PaK97] [PaC95] [TeH96] [voC92] [LiM94]. Some of the products produced during these efforts will be summarized in subsequent sections.

Overcoming the SBus bottleneck as described above centers on the OS. In the standard transmission control protocol with internet protocol (TCP/IP) and user datagram

---

<sup>1</sup> Tests conducted by Myricom achieved 44MB/s on transfers to the adapter and 66MB/s on transfers from the adapter which is documented at <http://www.myri.com>.



protocol (UDP) with IP implementations, the user's request for transmission is intercepted by the OS at which point the OS copies the data to kernel memory and context switches the operating environment to protected mode. The copied data passes through the OSI layers to the data link layer where the Myrinet Control Program (MCP) performs the transfer of data to the destination. The overhead required to implement this process severely hampers the realized bandwidth and latency. In fact, "loopback" testing (where messages are sent from one session layer port to another on the same machine) shows that the zero byte transfer latency associated with the Myrinet TCP/IP stack is .369 milliseconds (ms) as displayed in Figure 2-1 below. As a result, researchers developed new messaging algorithms that avoid much of the unnecessary overhead inside the standard methods. Myricom identified the problems associated with this method and supplied improved programming interfaces that allow Myrinet users to avoid the overhead. Their TCP/IP version is considered a "1-copy" implementation as described in [BoC95]. The support for "0-copy" transfers as recommended by the authors is displayed in the work of several researchers, which will be further explored in Sections 2.1.2 and 2.2.1.

#### ***2.1.1.1 First Generation Myrinet Switches***

Myricom's original commercial switch development is called the first generation as opposed to the second generation that is discussed in Subsection 2.1.1.2. The first generation provided support for links running at 640Mbps at distances up to 80 feet. The switch provided two processors where one performed the switching functions and the other adhered to the Myrinet specifications. The average path formation through the switch as provided by

Myricom is 300 nanoseconds (ns). This switch operated under Myricom's "LAN" definition as opposed to their "SAN" version.

### 2.1.1.2 Second Generation Myrinet Switches

The second generation switches support the 640Mbps-link speed and provide support for 1.28Gbps. The latter speed reduces the length of a link segment to 35 feet and under. In addition, the switches operate under Myricom's "SAN" standard with conversion logic that supports the "LAN" properties. As a result, a worm traversing a switch incurs a 100ns conversion delay<sup>2</sup>. The second-generation switch is the basis for the remainder of this research.

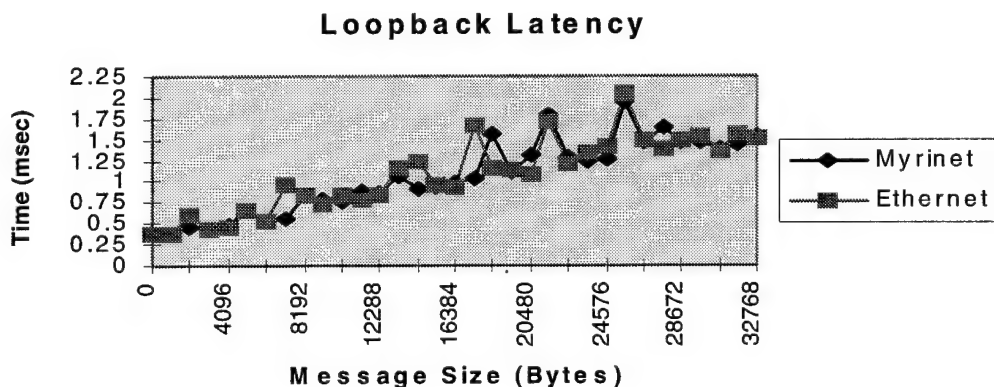


Figure 2-1 Loopback Latency for Myricom's TCP/IP Stack

<sup>2</sup> Information obtained from Mr. Chuck Seitz, Myricom.

### **2.1.2 Discussion of NOW Research**

The foundations of the Myrinet originated out of the Caltech Mosaic [SeB93] and the University of Southern California ATOMIC LAN [FeD94] research work which serve as the basis for this particular NOW environment. Further efforts to improve the corresponding NOW environments led to new developments in messaging. The efforts of the University of California at Berkeley (UCB) provide extensive support for NOW projects [AnC94]. Anderson, et al, defined the rationale for NOW environments, such as that presented in this research effort, by formalizing the support for shorter development cycles, improved cost-performance ratios, distributed processing power, and the need to improve communication protocols to support NOWs. Berkeley's message layering, Active Messages (AM) [voC92] [LiM94] initiated the research involving message layering in a NOW environment. Further discussion of AM and other messaging paradigms is provided in Section 2.2.1.

Beyond the message layering efforts, researchers extend the NOW paradigm to focus on pressing computational problems. Researchers at the University of Texas at San Antonio [ZhY95], for example, support the use of NOW environments for heterogeneous parallel processing use. In this scenario, a NOW provides a cost-effective solution for parallel problems, such as image processing, numerical computations, and data processing, while ensuring fair workload distributions. Berkeley's NOW World Wide Web homepage, *The Berkeley NOW Project* (currently located at <http://now.cs.berkeley.edu>), displays research efforts involving operating systems, file systems, and various applications. Additionally, research material from other works will be referenced as appropriate.

### ***2.1.3 Application of Past NOW Research to this Project***

As discussed above, the NOW research bodies attempt to cover computational challenges in various areas of parallel processing. The specific area of interest for this research centers on efficient messaging and routing to improve digital signal and image processing (DSIP). Few published reports provide specific insight into the application of NOW in support of DSIP. However, the research identified in the previous section does provide some areas that can be scoped to support DSIP. And, the remainder of this chapter will aim to identify areas of past research that might have applicability to supporting DSIP in a NOW.

## ***2.2 Ideology***

This section serves to lay the foundation for the research contained within this thesis. The message layering research to date will be summarized followed by a discussion of traffic analysis and traffic modeling. Finally, the issues as presented shall be tied into this research effort in the summary, Section 2.3.1.

### ***2.2.1 Presentation of Message Layering***

The base message layering paradigm exists in nearly every networked computer, that is, the standard OSI reference model implementing TCP/IP as the delivery method. TCP/IP provides a reliable transmission layering approach that when paired with a traditional OS, as the authors of [CIJ89] note, results in repeated copying of the user data and increases overhead to ensure the services guaranteed to the user are met. Clark, et al, [CIJ89] dispute claims that TCP/IP is solely to blame and that if properly integrated into an advanced layered

communications approach could provide better results. Despite this claim, Pakin, Karamcheti and Chien [PaK97] allude to the fact that TCP/IP's inability to provide gigabit transmission speeds for small messages requires new layering approaches.

As mentioned above, Berkeley's AM work initiated the line-of-research in message layering in a NOW environment [voC92] [LiM94]. AM brought to the research forefront the idea that the system designers should integrate communications with the end processors to efficiently reduce the overhead experienced in the typical layered approach [voC92]. Their implementation ignored the traditional view that the software should assume an unreliable transmission medium and effectively reduced vast amounts of repeated work incurred as a message flowed from one process through the message layers through the end node's message layers to another process [voC92]. The reduction in overhead corresponded to an asynchronous pipelined network that realized the flexibility of the modern communication network. In addition, the AM system derives its name from the process of its communication. First, a message is placed in the pipeline, freeing the sending processor to continue working, where each message riding the pipe contains a pointer to a "user-level handler" that maps to a process at its destination. Next, at its destination, the packet is removed from the pipe by the handler and passed to the appropriate post with little delay. Finally, all computing continues unless a sender is unable to place a message on the pipe at which time the system is blocked until the pipe is available [voC92]. One feature not implemented in this scheme, as discussed in [voC92], is a software buffering scheme. The lack of buffering serves as a potential bottleneck in a scenario where deadlocks are probable. However, the authors note that their communication process maps intrinsically with message passing system capabilities by which a system authorized interrupt mechanism is executed at

the arrival of a message. The effective result eliminates the need to provide complex messaging processors that drive up costs while reducing portability.

The University of Illinois (UIUC) sought to extend Berkeley's work which resulted in a new message layering protocol known as Fast Messages (FM) [PaC95] [PaK97] [LaC97]. Their goals included providing reliable, in-order communications while implementing a means to separate the system's processing unit(s) from the underlying network, thus freeing the system to maximize processing despite a blocked communication channel [PaK97]. Flow control and buffer management coupled with the underlying network's low BER provides the guaranteed delivery service. Pakin, et al, [PaK97] note the trend of programmable network interface processors, which includes Myrinet, that supports the ability to implement advanced service requirements at a lower level. Furthermore, FM achieves decoupling of the system processor and the network due to features implemented in their LANai Control Program (LCP) which resides in the Myrinet processor's memory space when loaded. Clark's, et al, assertion [CIJ89] that the traditional view of the network is not valid in gigabit-per-second network environments gains support in [PaK97]. Pakin, et al, note that in a Myrinet communications exist primarily as a message passing feature which further diverges from the traditional view of the underlying network. In fact, they formally identify the OS support needed in the Myrinet environment as merely a means to ensure configuration integrity is maintained (e.g., network interface initialized, interface assigned to only one process, etc.). This definition of OS support further justifies Clark's, et al, definition [CIJ89] of the OS's reduced role in network communications. In FM, user-level processes directly communicate with the network interface's send queue [PaK97]. Buffer overflow never occurs in a FM-

based Myrinet due to the credit flow control system. The authors of [PaC95] and [PaK97] claim that in-order delivery therefore is guaranteed.

As discussed, the authors in [PaL95] [PaK97] and [LaC97] provide support for improving the message layering in high-speed networks such as Myrinet. However, their work failed to address a few concerns. First, the authors fail to discuss how the process reacts to the rare bit error, as would occur with electrical disturbances that might occur. Second, what if a process's credits are all out and the system(s) that the process is communicating with is (are) not processing and not returning the credits? It appears that the authors expect the processing to continue at acceptable paces to neglect this scenario. But, on the sending process side, no communications over the Myrinet may proceed until the current process releases the network interface. The decoupling of the network and the processor will still prevent the system from entering complete deadlock. The concern here falls on the programmer to avoid setting up a parallel environment where one process is blocked waiting on another, which in turn leads to another that eventually cycles back. While the systems are free from deadlock, the Myrinet is effectively blocked for those systems. This rare occurrence is easily avoided if the programmer plans accordingly that result in this problem being of no real concern. Finally, their data merely shows ideal performance numbers which, are void of application specific uses. Despite these issues, the fact is that FM furthers the messaging paradigm beyond that established by the AM implementation.

Recently published information regarding another messaging implementation, the Real World Computing Partnership's (RWCP) PM [TeH96], claims to support an implementation to messaging that alleviates the problems with FM. That is, the authors state that PM provides context switching which keeps the sender process from blocking and that PM provides an improved flow control that ensures in-order delivery. The resulting increase in overhead associated with providing these services in their lower layer means that for smaller messages FM's latency is better. But as Figure 2-2 shows [TeH96] the latency of message sizes ranging from 200 to 1000 bytes in FM is almost 1.5 times greater than that of PM's. In addition, PM's peak bandwidth for messages larger than 400 bytes is nearly 1.5 times greater than FM's reaching a top rate of 32 Megabytes per second [TeH96] as displayed in Figure 2-3.

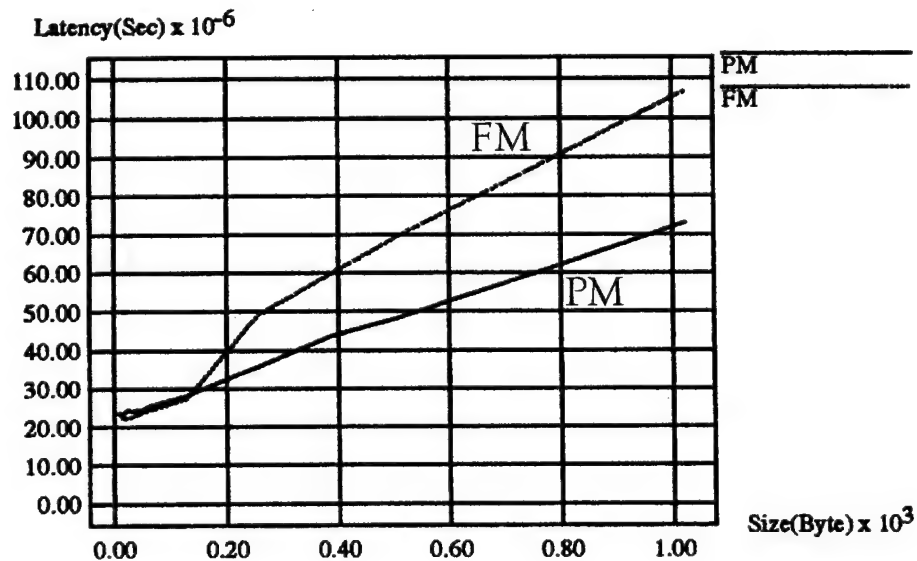


Figure 2-2 PM versus FM Latency [TeH96]



The question of which implementation is better now centers on the traffic characteristics for a target domain. Gusella [Gus90] identified 200 bytes as a size cap for typical data patterns in Ethernet traffic environments while data provided in [LeT94] correlates to an average of approximately 330 bytes. Most research efforts in messaging have shown the ability to provide vast amounts of bandwidth when optimizing for large messages [PaC95] [TeH96] [voC92]. However, Pakin, et al, [PaK97] supports a view that the problem lies in improving the bandwidth as well as latency in smaller messages since the larger case is well supported. This correlates with the goals of FM as addressed in [PaC95] and [PaK97].

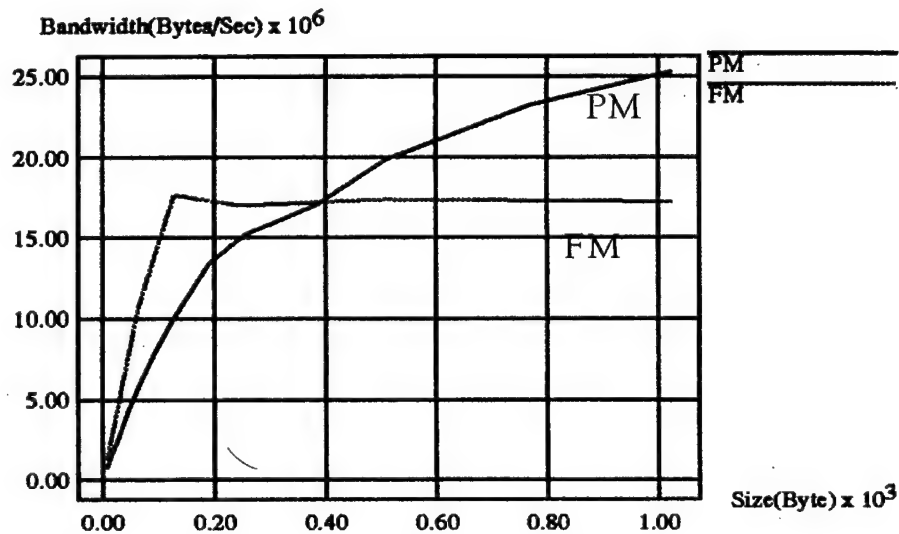


Figure 2-3 Bandwidth of PM versus FM [TeH96]

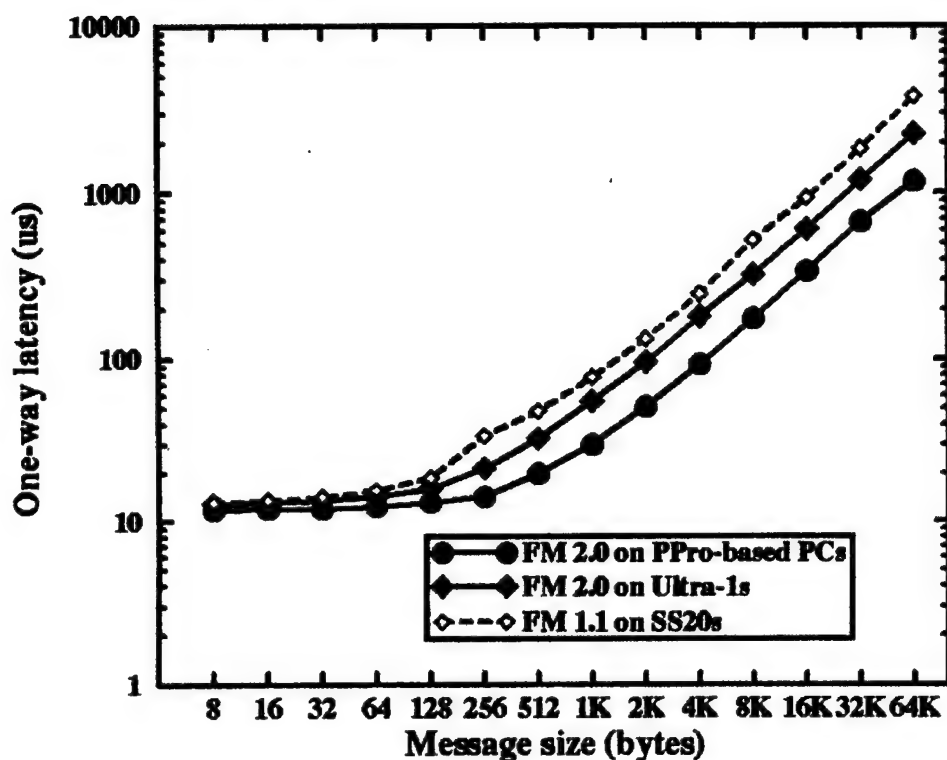


Figure 2-4 FM Latency Measurements for Sun Workstations and Pentium Pro Personal Computers [PaK97]

As displayed in Figure 2-4 [PaK97], these studies revealed that latency performance of a FM-1.1-based Myrinet exceeds that of the PM-based Myrinet on message sizes under 512 bytes for the Sparc Station 20 provided numbers. And, for the FM 2.0 case it outperforms PM for the message sizes provided in Figure 2-2. Figure 2-5 [PaK97] displays the corresponding bandwidth performance for the datum provided in Figure 2-4. Once again, FM 1.1 performs admirably for message sizes of 400 bytes and below as the values correlate to those provided in Figure 2-3. But, for FM 2.0, the two implementations perform equally in bandwidth for messages up to 1KB while FM's latency is about thirty percent better.

The issues outlined above lead to the obvious question of which tradeoff is better: support for variable sized messages or enhancing the common case. Thus, part of this research effort will expose some DSIP algorithms to traffic analysis to obtain a signature data description. The issues supporting traffic analysis and modeling in parallel processing are covered in Sections 2.2.2 and 2.2.3.

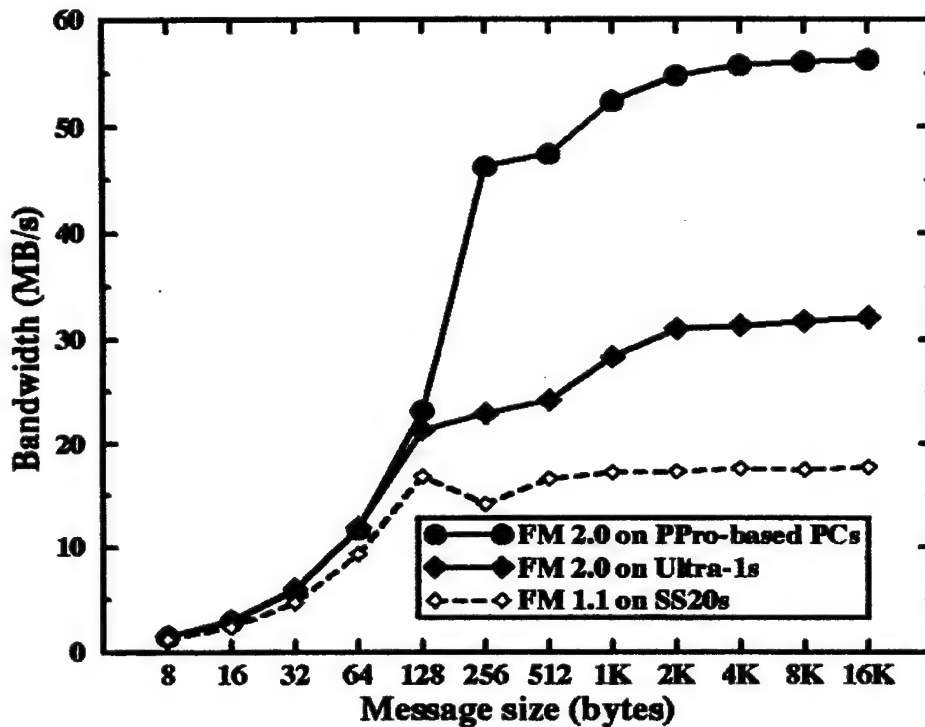


Figure 2-5 FM Bandwidth Measurements for Sun Workstations and Pentium Pro Personal Computers [PaK97]

## 2.2.2 Perspective in Traffic Analysis

Recent work in traffic analysis and modeling [LeT94] [WaM96] [PaF95] underscores a strong need to accurately quantify a network's traffic signature for modeling purposes.

These works show that standard assumptions regarding traffic patterns and the corresponding

modeling of such assumptions, such as the aggregation of Poisson traffic generators to model bursty traffic, might result in misleading data. Thus, through structured traffic analysis, researchers must justify their traffic modeling assumptions. Another area that these recently published works focus on is the failure to accurately model traffic via a Poisson process. Their work shows that a degree of self-similarity, defined as the level of structural similarity across a wide range of time scales [WaM95], remains evident across varying time scales. This directly correlates to bursty traffic which Poisson processes fail to accurately model due to the smoothing out of the traffic despite the aggregation of sources [LeT94]. Therefore, Chapter 3 of this research effort will discuss the process that this investigation employed to measure network traffic in an attempt to define a signature traffic pattern.

To identify a bursty structure, Leland, et al, [LeT94] suggest methods for identifying self-similarity (burstiness). For example, the authors provide pictorial graphs representing packet count per time unit versus the number of time units (called the variance-time plot). Other methods center on performing time-domain analysis where “R/S-statistics” or “periodogram-based” tools are used [LeT94]. Hu [Hu96] discusses burstiness of parallel traffic in a NOW (see Section 2.2.4).

The traffic generators for this research are based on soon to be published work at AFIT that provides detailed analysis of Fast Fourier Transforms (FFTs) in parallel computing platforms. The algorithms are subjected to different platforms to test the ability of a parallel environment, like a Myrinet, to support DSIP. Specifically, two FFTs are considered with varying optimizations implemented for each. The first algorithm, called Row Column or RC, takes a  $N \times N$  complex point image and performs the FFT using multiple processing nodes.

Initially, a single processor called the Controller or Con reads the image and scatters it in chunks to the remaining members of the group where a group's size is a power of two. The scattered chunk is equivalent to  $(N/p) \times N$  where  $p$  is the number of processors in the group. The members of the group minus the Con are called the Workers because their traffic patterns are slightly different resulting from the overhead performed by the Con to distribute and collect the image. However, when processing the one-dimensional FFTs, the Con is responsible for an equal workload.

A one-dimensional FFT is then performed by each processor on each row of its chunks resulting in  $(N/p)$  FFTs at each node. Next, the processors transpose the resulting data among each other in the chunks in to  $(N/p) \times (N/p)$  partitions. Performing  $(N/p)$  one-dimensional FFTs at each processor again processes the newly updated data. The resulting data is gathered by the Con where it's transposed to ensure correct ordering. For this research, a basic version of the RC algorithm, called RC Basic, and one optimized version, called RC Opt1, were monitored.

In addition to the RC class, a study was performed on a set of vector radix (VR) algorithms. This algorithm also required a Con to prepare an image and scatter it to the Workers. Each processor then performs their required FFT stages before producing  $\log_2 p$  communication stages. After all FFT stages are complete, the image is gathered by the Con where a permutation of the data is performed. Forthcoming work from AFIT details the intricacies of both algorithms, which is beyond the scope of this research. However, a minimal analysis of the algorithms allows shows that each produces a minimal value of data that must be communicated across the network. The parameters of interest to decipher this

minimum value are the size of the image and the number of processors in the group. Figure 2-6 shows the minimum size of data for both algorithm classes that must be communicated across the network assuming a 256x256 image where  $p$  is the number of processors. It is important to note that these values represent the data portion of a worm and do not reflect the overhead associated with communicating on a network.

2	786432	786432
4	1310720	1179648
8	1703936	1376256
16	2031616	1474560
32	2326528	1523712
64	2605056	1548288
128	2875392	1560576

Figure 2-6 Minimum Total Communication Size

### 2.2.3 Overview of Past Traffic Modeling Research

Closely related to and dependent on traffic analysis, traffic modeling receives equal debate among researchers. A Poisson generator's effectiveness to induce bursty traffic appears to fall short of the past assumptions [LeT94] [WaM96] [PaF95]. Therefore, a portion of this research work will focus on identifying the traffic signature as discussed in the previous section upon which simulations will be based accordingly. Paxson and Floyd [PaF95] discuss means for testing traffic models to attempt 95% confidence intervals which

would validate a Poisson generator's ability to model burstiness, for example. Their results show that only 15% of a sample (4.039 million packets) of TCP traffic is adequately modeled via Poisson generation as these types meet the exponentially distributed inter-arrival parameters.

However, further tests showed that the majority of the traffic fell into categories that reflect a heavy-tailed distribution, which corresponds not to Poisson generators, but to models that provide self-similar statistical properties. Further support of this self-similar nature is found in [WaM96] as their research effort further validated the inability of a Poisson traffic generator to create a truly bursty pattern despite varying degrees of aggregation (the number of sources presenting traffic to the network). Wang and McCrosky [WaM96] further correlated statistical analysis to models that best meet the properties of a single source generator and of an aggregated source generator. The general statistical background needed for self-similar processes is sufficiently discussed in [LeT94] [Pax95] [WaM96].

#### ***2.2.4 Discussion of Myrinet Routing***

Extensive work detailing routing in crossbar, worm-hole, and other similar routing switches provides a central theme: how much intelligence should the switch provide to best support routing features required by the customer? Obviously the concern will focus on cost versus performance. And, central to this research, the focus here is on the Myrinet wormhole routing switch. This relatively new device spawned a recent subset of background literature pertaining to these issues.

Before reviewing the literature, an introduction to Myrinet's routing implementations is needed. The blocking-cut-through (wormhole) switch provides 300ns path-formation latency as defined by the manufacturer's first generation switch specifications. Myricom considers the switch to be "perfect" in the fact that for any switching permutation, the number of packets may equal the number of ports. The switch provides only a transient state where state information of a worm is not saved. In addition, the switch does not provide any user programmable regions per the specifications. Additionally, the switch performs no cyclical redundancy checking so any errors are forwarded through the output port leading ultimately to the final destination where any errors are only checked upon complete arrival of a worm. This results directly from the wormhole routing where a worm is immediately forwarded to the corresponding output port upon decomposition of the header information. That is, determining the output port requires receiving a minimum number of flits that allows the remaining flits to be immediately forwarded. The switch bases its worm routing on source routing so that a switch's function is to simply steer worms. Furthermore, the switch provides multi-protocol support, low-error rates, and highly reliable delivery. With this underlying switch basis, the Myrinet routing scheme implements flow control through STOP/GO backpressure and the underlying cut-through routing support. The STOP/GO slack buffer support assumes link-by-link forward control where if the buffer fills to the stop point, a receiver generates the STOP control transmission. As a result, switches are designed with logic to perform the flow-control independently on each port. Thus, Myrinet avoids buffer overflow by controlling the amount of data transmitted from the forward end. The corresponding signal to restart transmissions occurs when processing of the buffered data drops the slack level below the go point at the forward buffer. The consequences to this



method result in each link requiring flow-control and in a worm possibly blocking across points (switches or hosts) in the network.

An additional feature of the Myrinet routing scheme provides network mapping to each system connected in a given Myrinet. The standard Myrinet Control Program (MCP) contains a unique 64-bit address, which serves to identify the "mapper" of a Myrinet. The mapper responsibility goes to the highest MCP address, and if this system should disconnect, then the next highest assumes command following the appropriate protocol for claiming control. The "mapper" scouts a Myrinet for valid, connected MCPs that respond to the scouting with an acknowledgment. Negative acknowledgments appear when a scout assumed a particular host's attachment at an address, but the responding host did not meet the assumed identifier. A scout to a port that receives no form of response is assumed to be a switch-to-switch (S2S) connection. The mapper tests the S2S connection via embedding appropriate routing data into the header of a scout that, once stripped at the corresponding hops, is returned to the mapper. Thus, S2S connections and recursive tree structures may be identified. Failure to receive the scout back at the mapper indicates that no attachment is present at that port. Furthermore, store-and-forward multicast results from sending data to a host whom, in turn, routes the data to the next hop. Broadcast and unicast features are typical in this routing environment. Myricom's planned expansion of the switch proposes a cut-through-multicast that establishes a tree-based multicast route via storing entries into a multicast table inside the Myrinet switch. Each entry would correspond to appropriate replacement routing information inside a packet's header (e.g., outgoing ports). Finally, Myrinet's specifications provide for coexistence between customer developed MCPs and the

delivered MCP. However, Myrinet assumes the customized MCPs adhere to the “Myrinet Participation Requirements” standard.

With this basis for the Myrinet routing and switch, the research potential in this simplified method for “steering packets” exists based on two fronts: 1) identifying features to upgrade the switch’s functionality that meet the attractive cost goals of the Myrinet or 2) maximizing the available hardware/software combinations to identify new means for enhancing the quality of service provided by the Myrinet. The latter option provided motivation for the work published in [VeL96] where the authors display the ability to support reliable multicasting based on UIUC’s Fast Messages. This multicast implementation provides support for what the authors identify as the two critical issues for a reliable multicast on FM for Myrinet: 1) defining the optimal spanning tree and 2) providing reliable delivery at the MCP (or, LCP in the case of FM) level thus avoiding the high overhead experienced at higher message layers. Interestingly, the authors claim that the interconnection of switches allows the arbitrary topologies to be viewed as “one large virtual switch” due to the crossbar design. The resulting simplified network allowed the authors to arbitrarily define the optimal tree structure. However, the need to identify the level of support based on reduced average latency or maximized throughput led the research work to base the tree on a binary structure to best offset the opportunity costs. Ultimately, the authors back a congestion avoidance method as congestion control suffers from its reactionary methods where the high-speed network delivered packets prior to the control mechanisms effectively reducing the transmission rates. Their final version provides a credit-based scheme where a centralized credit manager oversees the distribution of credit resources

resulting in a dynamic scheme where one credit correlates to one fragment's distribution to all specified end nodes.

A subsequent published article [GeP96] addresses the same front as the previous research effort by restricting the domain to the Myrinet switch without modifications. In addition, the authors address the second front as well by describing how modifying a wormhole-source-routing switch could support efficient multicasting. Their focus aimed to identify protocols for achieving deadlock free, reliable multicasting on both fronts. The latter front begins with a base structure similar to Myricom's switch. Thus, a general wormhole LAN is used and supported by a switch similar to a Myrinet switch. In addition, receiver based pressurized, flow-control is implemented, as well as, sender based routing to specify the path for a packet. The authors of [GeP96] liken this environment to a very reliable, very high-speed conventional LAN with improvements in latency gained from the underlying switch. However, as previously identified, multicasting falls short of the traditional LAN's service. The authors describe Myrinet's native multicast as suffering from the simplicity of the switch design which exposes the network to excessive delays and even deadlocking, not to mention the inability to guarantee "total ordering." The excessive delays are enumerative on a case-by-case basis depending on the number of nodes involved since a sender is required to replicate a message for each system in the multicast group. Again, this requirement stems from the fact that the switch does not provide hardware support for multicasting. Interestingly, the authors state that the method described in [VeL96] provides the neglected total ordering but their version suffers latency problems while exposing the entire scheme to a central failure point. Thus, an extension to their scheme would provide

the necessary fault tolerance, but again serving to only increase the potential latency problem.

With the noted problems in the switch, the authors in [GeP96] base their recommended switch services on the Myrinet implementation but note that no formal support is provided by Myrinet. In contrast to services provided by conventional networks, switch-level multicasting in networks based on a simplistic wormhole switch, like Myrinet's, suffer from back-pressure through an extended "tree" structure, from an inability to map destinations inside the switch, and from deadlocking as noted by the authors. Thus, their implementation focuses on these three areas but does not address Myrinet's proposed routing table support as described previously.

In addition to the switch-based approach, Gerla, et al, [GeP96] provide a dynamic source-routing-based methodology. However, the authors opt not to backpressure but rather to drop links that cannot fully buffer a worm through an acknowledgment scheme. At high-loads, the resulting communications structure appears to mimic a store-and-forward network. Furthermore, a single message might exceed the buffering capability completely. The resulting messages incur excessive overhead to break up the original message into acceptable chunks. The authors note that in [VeL96] a solution to this problem was to allow the host's DMA region to accept the overflow of the Myrinet's buffer region. In this fashion, the resulting bus access time for the overflow amount must not exceed the original time to break up the multicast message. Without formal verification, it appears that a multicast to a large group might not actually suffer from the overhead required to break up the message, especially if the DMA region is accessed at several intermediate nodes. But, the claim in

[GeP96] is that under normal loads a message will seldom overflow. However, the authors fail to define "normal loads" and only provide charts with loads ranging from three percent to twelve percent. Finally, their claim for deadlock prevention follows from the method describe in coordination with adherence to a few transmission rules to ensure that a Myrinet's buffer does not deadlock.

Finally, the authors of [GeP96] note that their investigations centered on two multicast structures. The first, a "Hamiltonian circuit" actually forms a subset of the second, a rooted tree structure supporting various degrees of fan-out. Their reported simulation results showed varying performance based on network loads. Thus, actual implementation might fall back to a case similar to [VeL96] where a binary tree provided the optimal balance.

Additional research conducted on Myrinet's high-speed network involved identifying performance enhancements using time-out reset mechanisms in a NOW [Hu96]. Hu provides simulation results supporting a time-out-reset mechanism for wormhole routing in a Myrinet-based NOW. As used in traditional networks (store-and-forward), the time-out prevents deadlocks and is typically a large value. However, the use of time-outs in wormhole routing is not as straight forward where an unreasonably small value may make the worms just repeat a cycle of try and time-out [Hu96]. The same can be said for unreasonably large values that do nothing more than guarantee the deadlock disaster. But, the optimal value can be shown to improve the blocking probability in wormhole routed networks [Hu96]. Hu's implementation of the time-out algorithm results in a timer for each worm head that reaches a switch and a corresponding time-out that occurs when a threshold

value is exceeded. The resulting back-flow of resets continues until the originating host is notified upon which it either 1) stops the on-going transmission (if processing of that particular worm is continuing) and re-queues the worm or 2) it simply re-queues that particular worm [Hu96]. In either case, the network must be out-of-order delivery capable. Hu's simulation environment uses uniformly generated traffic on all hosts with destinations chosen randomly based on distribution probability from a corresponding network configuration. So, for the network provided in Fig 2-7 [Hu96] (3 x 3 torus network) the resulting destination distributions are assigned according to a probability of equal distribution across hop distances of two, three, and four.

In addition to that provided in Figure 2-7, larger network configurations are used and the corresponding distributions are scaled accordingly [Hu96]. The simulations assume Poisson generation for source worm creation with corresponding lengths exponentially distributed. Hu's simulations also follow Myrinet's pre-defined slack buffer space (the area between the low- and high-threshold regions) of 16 flits<sup>3</sup>. Furthermore, Hu explores variations of buffer space in relation to the time-out mechanism<sup>4</sup>. The results show that for buffer sizes larger than the time-out period throughput uniformly rises from 16 MB to a peak of 16.5 MB and declines as the time-out value is increased.

---

<sup>3</sup> A flit is defined based on the per clock cycle data transfer size in a wormhole routed network.

<sup>4</sup> For simplicity purposes, the buffer size was defined using the empty to low-threshold region. In actuality, the buffer is used to facilitate the propagation delay such that the defined size is added to twice the size needed for storing the in-transit data (propagation delay).

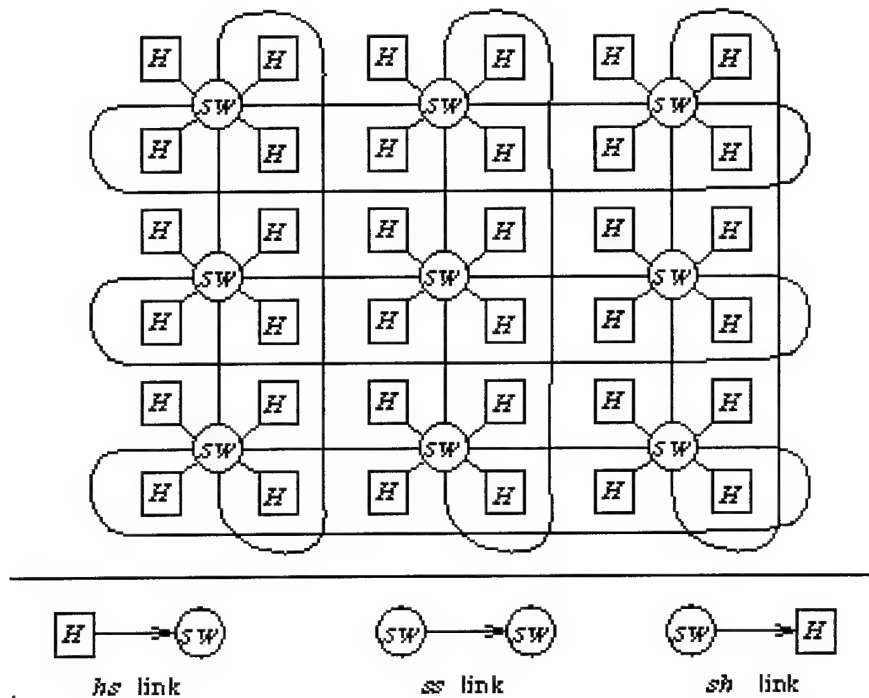


Figure 2-7 3 x 3 Torus Network [Hu96]

As Hu notes, this results from the buffer beginning to fill up with a worm but quickly freeing up the space as the time-out value relatively keeps the buffer free. However, progressing down an infinitely varying continuum of time-out values, the network encounters the head of line block problem as the buffer size varies accordingly and up to a theoretical infinite value in spite of adequate congestion control. Hu defined the case where the wormhole routed network with an infinite buffer and an infinite time-out value acts as a virtual cut-through network employing input buffering [Hu96]. Hu notes that with sixty percent link utilization and uniformly distributed traffic the maximum throughput achievable by the virtual cut-through network in a 3 x 3 torus configuration is 21.6 flits per cycle. But, through the use of an optimal time-out, Hu's simulations displayed throughput rates greater than these that produced values ranging from 22 MB to 25.5 MB with increasing worm

lengths or as propagation delay is reduced. However, simulation results also pointed to the problems associated with scalability of a wormhole routed network. The results showed that the optimal time-out value varied only slightly as the network size increased. Furthermore, the efficiency of network utilization or, as Hu defines, the “ratio of maximum throughput to maximal throughput” declines as the size scales up. Simulation results showed that only fourteen- percent efficiency was realized for a 7 x 7-torus network versus forty-seven percent for the 3 x 3-torus network.

In addition to buffer size, Hu investigated the optimal time-out value. Simulation analysis was performed on network size, varying time-out values, varying network loads given a fixed worm length, varying propagation delays, and static network configurations with varying worm length. As Hu notes, the penalty paid for not implementing the correct time-out value is disproportionate depending on the degree of the error as noted in the varying network size case. At the optimum time-out value of 40 cycles for a worm length of 50 flits, the throughput reaches approximately 26 MB but drops as the time-out value exceeds that point whereby throughput drops to 24 MB at 100 cycles (2.5 times the optimum). However, the throughput for time-out values smaller than the optimum decays much more rapidly as seen by the fact that at half of the optimum (20 cycles) the throughput reaches only 24 MB [Hu96]. At the point of immediate time-out, the throughput achievable is only 17.5 MB [Hu96]. Under network loads of fifty percent using a static network configuration and static worm length (100 flits), the optimal time-out value is one cycle where throughput is 17 MB [Hu96]. An increase in the time-out value for the given parameters is only 14.25 MB for time-out of 100 cycles and reaches only 8 MB for the 2000 cycle case [Hu96]. For longer worm lengths, Hu’s simulations showed that as worm length



increased the optimum time-out value decreased. This fact is exemplified in Figure 2-8 [Hu96].

Hu concluded that for the bursty nature of a LAN, such as the parallel-processing environment using a Myrinet, the high bandwidth needs correlate to the justification of longer worms which reaps the associated benefit of high throughput. This is in spite of results that showed the low delay benefit of smaller worms in a lightly loaded configuration. Furthermore, the penalties associated with the wasted bandwidth associated with a time-out mechanism are justifiable if it can be shown that a network exhibits high-loads (forty percent or greater). But, the value in finding the optimal time-out is high when considering the propagation delay, the network degree, and the size of the transmitted worm.

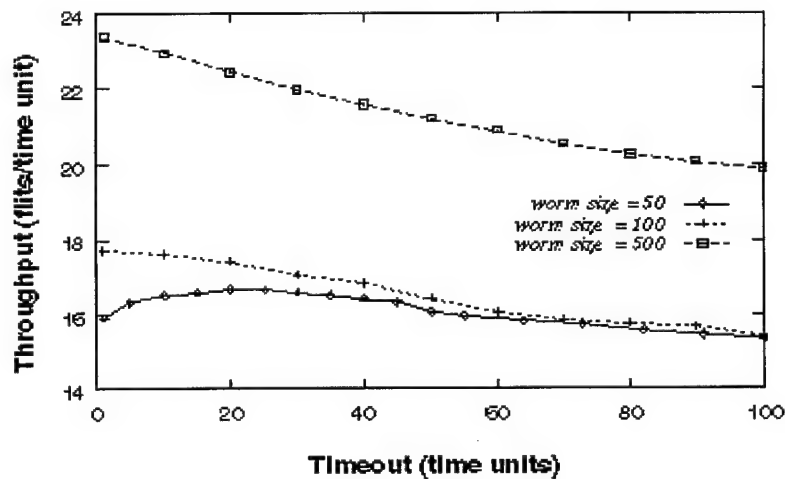


Figure 2-8 Timeout and Throughput for Varying Worm Length [Hu96]

## **2.3 Summary**

With the emergence of high-speed LANs, the parallel processing community reaps the benefits of low-cost message passing environments. The availability of off-the-shelf components allows the designer to employ the industries latest technological advancements. In addition, the economics of volume productions place configurations such as NOW in a high-leverage position over traditionally developed parallel processing environments where costs often reach millions of dollars. Starting with the ideas outlined throughout this background review, this research effort aims to further the field of parallel processing in a NOW environment.

### **2.3.1 Foundation of this Research Project**

The core of this project centers on Myricom's Myrinet and the previous research involving the associated NOW environments. Areas of direct emphasis include UCB's, UIUC's, and RWCP's message layering research. Ongoing research at AFIT focuses on identifying parallel processing support for DSIP applications through comparisons of Fast Fourier Transform algorithms involving platforms such as the NOW used in this research. In addition, that research project will explore the use of UIUC's FM protocol in comparison to Myrinet's base protocol. Traffic analysis will be conducted during the implementation of these algorithms to assist in the definition of a traffic pattern for use in this project (explicit methodology will be defined in Chapter 3). In return, this research effort will explore support for these algorithms through varying network parameters such as those used in Hu's research [Hu96].

### ***2.3.2 Divergence from Past NOW Research***

The issues displayed above imply that an obvious divergence from past research work is to demonstrate support for DSIP in a NOW versus the traditional stove-piped massively parallel processor (MPP) based environment. Thus, this work shall further the field of NOW study by demonstrating the ability to support DSIP through scaleable NOW configurations. Some issues provided in the background research don't fully map to the DSIP environment as outlined previously. Therefore, through traffic analysis, modeling, and simulation, this research work will highlight specific DSIP applicability for which the reviewed work did not show direct support. Specific areas of focus will be further defined in Chapter 3.

### ***III. Methodology***

#### ***3.1 Introduction***

This chapter presents the methodology used for this research project to create a simulation model of a Myrinet and to analyze the performance of this network. In [Jai91], the author defines three means for obtaining computer network performance data: 1) analytical, 2) simulation, and 3) measurement. To more precisely obtain measurements, Jain recommends the use of at least two of the methods simultaneously to further substantiate the data. For this research effort a combination of simulation and measurements will be performed with the simulation acting as the primary research tool. The remaining sections of this chapter deal with defining the Myrinet simulation model. First, the rest of this section introduces the modeling environment. Section 3.2 discusses the design methodology while Section 3.3 lays out the process of verification and validation of the model.

##### ***3.1.1 Network Architectures***

The Myrinet environment was introduced in Chapter 2 and is modeled through the use of BONEs DESIGNER as described in Subsection 3.1.2. Scaling the number of systems as presented in Subsection 3.2.3 developed the actual network configuration sizes ranging from 2 to 128 by powers of two. The architectures are protocol independent and based on the second-generation switches. Therefore, all links run at speeds of 1.28 Gbps. For the purpose of this research, the simulations emulate the base Myricom TCP/IP protocol stack implementing parallel processing communications traffic via message passing. Subsections 3.1.3 and 3.1.4 discuss the traffic patterns used in the actual simulations as this research aims to use actual measured parallel processing traffic. All architectures were designed using

Myricom's 8-port Myrinet Switch. Each topology was tested using loads that simulated measured parallel processing data transfer rates using data collected from an active Myrinet LAN as enumerated in Figure 3.1. The maximum achievable load correlates to the overhead burden of using TCP/IP. Chapter 2 showed that the minimum latency achievable during an MPI roundtrip test was limited by an overhead that was measured to be 369ms for a data size of zero bytes. This value was measured on Sun's SPARCstation Ultras as displayed in Figure 2.1. This workstation was used due to its advanced technology and its availability for measurements of real-time data. However, these values correlate to the overhead encountered while running in a multi-user environment. In the Myricom protocol stack, the overhead is incurred until the communicating process is granted DMA authority to move a worm from the workstation's main memory to the Myrinet adapter's SRAM. Therefore, the validation of the model employs tests that incur this overhead but the actual simulations ignore this overhead to alleviate the need to quantify the randomness of this variable.

### **3.1.2 BONeS Designer**

The development of BONeS Designer, or simply Designer, by the Alta Group of Cadence Design Systems, Inc. provides an ideal off-the-shelf product for computer network simulation. Designer's built-in "primitives" support a hierarchical approach to developing detailed simulation models. Designer's graphics based environment allows the user to link blocks that provide the needed functionality to analyze network communications data. The integrated development approach supports interactive debugging, discrete-event simulations, and flexible design of advanced primitives that are the building blocks of advanced architectures. Furthermore, it supports high-level and fine-grain development. High-level

views provide representations of network traffic flows such as end-to-end communications where a source creates a communications data structure, a link mechanism carries the data structure, and a destination receives and processes the data structure if necessary. Fine-grain abstractions allow models of internal components down to the gate level. The data analysis features of Designer allow the user to define exact metrics and to specify the correlating points of interest.

### 3.1.3 *Traffic Analysis*

The process of modeling a parallel-processing environment that implements message passing was supported through analyzing actual traffic patterns. The specific details involved recording the source address, destination address, the size of the data portion, and a time stamp when the worm was identified as a TCP/IP packet in a Myrinet environment. To accomplish this task, a non-intrusive software trap, *tcpdump*<sup>5</sup>, was loaded into the memory space of four Sun Sparc Ultra workstations. The data captured was localized according to the specific Fast Fourier Transform (FFT) algorithm. Figure 3.1 lists the different captures performed. During the data capturing five variations of FFT algorithms targeted toward parallel processing were monitored. These specific applications were targeted to attempt to define an ideal traffic signature for these parallel applications. In the next subsection traffic models are defined that support the actual simulation patterns. The traffic monitored over the Myrinet helps to define the actual patterns for the targeted application domain, digital signal and image processing (DSIP).

---

<sup>5</sup> Obtained from the Lawrence Berkeley National Laboratory's Internet Traffic Archive via <ftp://ftp.ee.lbl.gov>.

Capture	FFT Algorithm	Num. Of Nodes	Matrix Dimension (n x n)
RCB_256_2	Row Column Basic	2	256
RCB_256_4	Row Column Basic	4	256
RC1_256_2	Row Column Optimization Number 1	2	256
RC1_256_4	Row Column Optimization Number 1	4	256
VRB_256_2	Vector Radix Basic	2	256
VRB_256_4	Vector Radix Basic	4	256
VR2_256_2	Vector Radix Optimization Number 2	2	256
VR2_256_4	Vector Radix Optimization Number 2	4	256
VR4_256_3	Vector Radix Optimization Number 4	2	256
VR4_256_4	Vector Radix Optimization Number 4	4	256

Figure 3-1 Myrinet Traffic Captures

As outlined in Chapter 2, research shows that traffic patterns across a network exhibit differing degrees of self-similarity, defined as the level of structural similarity across a wide range of time scales [WaM95]. As noted in [LeT94], the failure to accurately model a network using Poisson generators potentially leads one to false assumptions regarding the performance of a network. Thus, in accordance with Leland's, et al, [LeT94] variance-time plot method, this work captured the traffic statistics for the FFTs under investigation. To accomplish this task, the *tcpdump* program's output was analyzed by feeding the data into a script that counted the number of worms transmitted per a given time unit. The degree of the

time unit directly affected the ability to plot the data to minute scales. However, data was collected for each FFT at .0001, .00025, .0005, .00075, .001, .0025, .005, .0075, .01, .025, .05, .075, and .1 second intervals. Chapter 4 and Appendix A discuss the results of the traces.

#### **3.1.4 Traffic Modeling**

The traffic analysis discussed in Subsection 3.1.4 is used to feed a traffic generator to emulate the network dynamics. Two steps were taken to identify the generator best suited to model the network. First, the primitives provided by Designer were reviewed to determine if one, or more, applied to the model. Two primitives, the Arbitrary Pulse Train and the Bursty Pulse Train (Expon, Geom), met the needs of this research and are used in the simulation tests. The former provides the functionality to accept a user-defined series of Trigger signals, which supports the simulations that use the actual network traffic as input. The latter fits the need of producing the self-similar traffic streams as identified in Chapter 4 and Appendix A.

Next, the data was fed to a utility that produced the average number of worms generated during a given window period. The window size ranged from .05 to 5 second in .025 intervals. It was determined that the 1 second interval best captured the dynamics of each algorithm based on finding the point where the number of messages within the windows stopped increasing. In addition, the average size and the variance of the size for the worms were calculated. In turn, these statistics, provided in Appendix A Section A-3, are scaled to produce equivalent values for the topologies consisting of eight or more systems. These values assume that the size of the image will scale accordingly. This is due to the fact that as



the number of processors increases, the size of the communicated worms would decrease according to the algorithms as discussed in Chapter 2.

### **3.2 Design**

This section describes the simulation model and the issues behind the decisions made that affect the model. The first subsection justifies the process of simulation supported by actual system measurements. Next, Subsection 3.2.2 defines the assumptions made in the model. The actual construction of the model is detailed in Subsection 3.2.3 and the parameters for the model are defined in Subsection 3.2.4. The Measurements Subsection, 3.2.5, discusses the performance metrics of interest while the actual simulation experiments are identified in Subsection 3.2.6.

#### **3.2.1 Method**

The purpose of this simulation study is to present analytical data that can be used to support the performance of DSIP parallel processing algorithms in a Network of Workstations. Topologies consisting of 2, 4, 8, 16, 32, 64 and 128 nodes have been studied to compare the relative performance of each in terms of throughput and latency. The actual results of the simulations are discussed in Chapter 4. A simulation model was chosen as the primary tool for providing the analytical data used in this research. The simulation method as opposed to Jain's two other approaches for obtaining performance data [Jai91] was identified as the approach that best supported this research effort. The simulations allow for far greater resources to be modeled than the analytical approach, which becomes an obstinate problem for the size of the networks in this research. In addition, measuring the actual

networks was not an option due to the limited resources available. Therefore, measurements were conducted to produce a scaled model of a Myrinet's performance. Prior to taking the actual FFT traces, measurements were taken on two directly connected systems that represent an environment where a system encounters no contention to send to a destination node (see Section 3.3 for a discussion of the tests). Measurements showed the highest throughput 17.8 megabytes per second (MB/s) that a single communication thread could obtain using MPI as the messaging platform. The problem with taking these measurements at face value is that variability is encountered in the operating system's interaction with the communication processes. Therefore, during the validation phase (see Subsection 3.3.2) tests were conducted that attempted to quantify the overhead. In addition, measurements were performed on two- and four-node workstations connected to an 8-port switch to produce the FFT traffic signature as defined in Subsection 3.1.4. The simulation method used in this study is discrete-event.

As previously stated, the modeling package Designer was used to create appropriate models to support the analysis of the Myrinet topologies. The flexibility of Designer supports a top-down approach to model creation, which was used throughout the development phase. At the highest level, the system view allows the interconnection of blocks representing the processing nodes, the Myrinet switches, and the interconnection links. Appendix B provides the Designer block diagrams for each of the topologies used during this research.

Within a configuration, the switches are divided into the input port module, the switching fabric module, and the output port module. Each of the output port modules is

further subdivided into an arbitration unit, a multiplexing unit, and a de-multiplexing unit. In addition, the workstations within a configuration are further divided into a Source process block, a Destination process block, and a Myrinet Adapter process block. The three main components of the system are further developed via a top-down fashion that is described in Subsection 3.2.3.

### **3.2.2 Assumptions**

Simplifying assumptions were used during the model development phase to narrow the variability in the parameters of interest:

1. Switches are error free: Due to the manufacturer's claim of extremely low error rates, the validation process would require monitoring actual switches for a period that covers 125E6 MB. Thus, this simulation study does not inject random errors into the switch. Errors of this type would be detected on an end-to-end basis within the messaging layers.
2. Links are error-free: The relatively short distance (6m) is assumed to be manageable to alleviate faults. In addition, this research is primarily concerned with the operation of the switches, which precludes the need to inject errors. The messaging layers in the workstations would handle such errors.
3. Messaging layer limited to the manufacturer's TCP/IP stack: The availability of Myricom's specifications and documentation eases the effort to interpret the communication process and overhead. In contrast, the availability of other messaging paradigms such as FM or PM is limited in the sense that binary executables are not readily available. In addition, the documentation is not fully developed to the extent that Myricom's is which limits the potential for interpreting the overhead involved in these other communication processes.
4. Main memory in workstations available as needed: The workstations generate packets at a rate identified during the traffic analysis. It is assumed that main memory will always be available to store a packet awaiting direct memory access (DMA) transfer to the network interface. Similarly, it is assumed that main memory will always be available for the network interface to DMA a packet to main memory upon receiving the trailer flit from a sending node.

5. Network nodes are always available: This research effort is concerned with the performance of the network components in relation to the routing and connectivity options presented within. The injection of random node errors (switch or workstation) would correlate to testing the messaging layer and its robustness in the face of errors.

### **3.2.3 Construction**

The top-down approach to designing this model began with the construction of the high-level system view. Non-functional blocks representing the switches, workstations, and links were placed to provide a graphical view of the flow of data through the network. Each of the three different components was then broken down into the sub-functional units as listed in Subsection 3.2.1.

#### **3.2.3.1 System Level Construction**

At the system level, the different configurations represent the varying methods for interconnecting workstations and switches. The main criteria for developing the *fat-tree* configurations is that 1) each node is active and 2) each node can fully communicate with every other node in the network via the necessary inter-connecting links. The performance in relation to each of the various FFT loads is of primary concern. Chapter 4 provides analysis of the performance that each load achieves. As the figures display in Appendix B, there are specific parameters that are common throughout the system and are listed in the heading "COMMON ARGUMENTS." Parameters within designer are labeled with a "P" while memory variables are displayed with an "M." Parameters and memory variables within designer can be local, offset by an encompassing box around the "P" or "M," or exported to a higher block which do not have an encompassing box. Subsection 3.2.4 further discusses the parameters used in this simulation.

### 3.2.3.2 Switch Level Construction

The switch has been constructed to follow Myrinet's design as closely as possible (refer to Chapter 2 for detailed design information). The main concern is to incur delays for:

1. Assigning an input stream to an output port: 6.25ns upon arbitration deciding on the next stream.
2. Moving a flit from the input port to the output port for 6.25ns according to Equation 3.1 for average path latency:
3. Processing the SAN-to-LAN conversion: 100ns.

$$\frac{1}{\text{switch processing speed}} = \frac{1}{1.28Gbps} \quad (3.1)$$

The switch operates independent of what the next node is on any of the ports. That is, the switch was designed to facilitate switch-to-switch connections as well as switch-to-workstation connections. In addition, the model emulates Myrinet's link-by-link flit buffering in both the Switch and the Adapter. The switch model accomplishes this through STOP-GO flow control inside each input port.

Since the switches were also designed following a top-down process, the switch can be configured to support varying numbers of ports. For the purpose of this research only eight port switches were used. This was accomplished by aggregating four, one-port blocks together to form an  $8 \times 4$  block. Then, two of these blocks were then aggregated together to implement an  $8 \times 8$  block. This block represents the actual switching fabric within a switch.

To complete a Myrinet switch, the input interpreters are added for each port. A similar process can be followed to create larger or smaller switches that conform to Myrinet's designs.

### ***3.2.3.3 Workstation Level Construction***

The workstation's construction followed the same top-down design approach by starting with a black box view and, filling in the functionality within it as appropriate. In addition, the workstation allows the modeling of varying functionality. For example, a workstation may use the measured timing analysis of the operating system's overhead to interject additional variability into the performance. In contrast, the workstation may simply take traffic distributions that are measured after the operating system has delayed a process from initiating its communication routine. This feature allows the measurement of the network's performance for worms initiating just above the system bus that are tracked until the worm is sent across the system bus at the destination. Each of the variants is easily accomplished by simply replacing the "Advance Source Process" block from with the desired functionality. Also, the delay resulting from a worm traversing the Myrinet Adapter is included in the Workstation. The functions of the workstation are decomposed into three components: 1) a sending process called Source, 2) a receiving or sink process called Destination, and 3) the network interface process called Myrinet Adapter. In addition, the system's input/output bus is accounted for within the Workstation.

The sending process is used to generate packets, called worms, as opposed to messages. This simplification was made because of the measured traffic patterns monitored

only individual worms and not full session traffic patterns. Also, the measurement of worms allows this research to concentrate on the key concern: efficient uses of Myrinet resources to better support the DSIP processing domain. The sending process generates a worm by first setting the data size and secondly setting the destination address. Data sizes are limited to 8 kilobytes. For the actual performance simulations, the destination addresses are chosen based on the traffic patterns observed during actual traffic analysis reviewed in Subsections 3.1.3 and 3.1.4. After establishing the worm, the source then notifies the network interface of a pending packet. If the interface has space available in its *SRAM* region, it signals the source to release the worm. The network interface then begins its sending responsibilities as listed below upon receiving the worm as detailed below:

1. Delaying a worm until the output port is ready.
2. Translating a destination address into a Myrinet route.
3. "Flitizing" packets that are DMAed into the network adapter.
4. Transferring a packet (flit-by-flit) to the next node.
5. Delaying the flit transfer process in response to the "down channeled" STOP signals (and starting the process in response to the corresponding GO signals).
6. Freeing the *SRAM* region upon sending the final trailer flit.

Upon receiving a GAP symbol from an up stream node, the network interface process checks for available flit space in the STOP/GO flow control queue. If space is not available, then the process sends a STOP signal. Otherwise, the network interface accepts the incoming flits and performs the following functions:

1. Storing the flit in the slack buffer.

2. DMAing the packet (upon receipt of the trailer flit) to main memory with delay to Equation 3.2.

$\frac{\text{Size of packet in Bytes}}{\text{Transfer Rate in MB/s across I/O Bus}} \quad (3.2)$
--

### 3.2.4 Parameters

The parameters listed below are divided to show those that apply to the Myrinet (including the Myrinet Adapter process) those that apply to the workstations above the Myrinet adapter, and those applying to the simulation environment.

#### 3.2.4.1 Myrinet Parameters

1. Maximum SRAM Size: 256KB used for both receiving and sending sides where the network interface is responsible for managing the space provided to both sending and receiving functions.
2. Size of Region above the Stop Line: 32B which applies to the STOP/GO flow control; this value represents the potential in-transit data throughout the route per the Myrinet specifications.
3. Size of Region below the Go Line: 32B used in the flow control process; this value is used to initialize the operating region.
4. Size of Hysteresis Region: 16B which applies to the receive processing performed in the network interface. This represents the optimum, operating region in the STOP/GO flow control.
5. Number of Ports: 8 the number of ports on a switch which is typically the same for each switch in the network but can be set individually. The value includes interconnection links as well as attached systems. Per Myrinet's designs, this value is expected to vary from 4 to 32 depending on the switch in use.
6. Average Path Latency/One Flit Delay: 6.25ns, as defined in Subsection 3.2.3.2, this value is the time for a flit to traverse the path from input- to output-port.



7. Link Length: 6m which is fixed for each link used in the network. This value is used within the *Link* Block to compute a propagation delay according to Myricom's defined percentage of the speed of light across the link as listed in Equation 3.3.

$$\frac{\text{Length of Cable}}{\text{Percent Obtainable} * \text{Speed of Light}} = \frac{x}{.6 * 3E8} \quad (3.3)$$

8. Myrinet Route Table: *filename* a pointer to the file containing the route table for generating the node and port combination for sending a message across the network. The route table defines the path(s) a flit will travel to traverse from a source to a destination. An example of an actual table used in this research is provided in Appendix C.
9. Adapter Flit Creation Delay: 6.67ns which is obtained from the Myrinet documentation which states that the Sun Myrinet Adapter operates on each edge of a rate equivalent to three times the SBus rate, 25MHz. However, this parameter can be used for any I/O bus as displayed in Chapter 4 where values corresponding to a PCI bus are used.
10. Round Robin String: 6 5 7 4 3 0 2 1 the order of arbitration within a switch. Chapter 2 discusses the arbitration process within the second generation Myrinet switches.
11. Conversion Latency: 100e-9 the time in seconds that each worm encounters for the SAN-to-LAN conversion inside a Myrinet second generation switch as discussed in Chapter 2.
12. Number of Input Ports: 8 the number of ports that a switch has. This value was exported to the highest system level for the simulations used in this research but, the value can be used on a per switch basis to support scaling the size of a switch.

#### 3.2.4.2 Workstation Parameters

These parameters are divided into two subsections to differentiate between them. The first subsection identifies the common parameters used by all workstations. The final subsection lists the parameters used in the workstations that use statistical representations defined by the process detailed in Subsection 3.1.3. This is opposed to the workstations that use the actual traffic traces identified in Subsection 3.1.3. These workstations do not have

any parameters beyond those identified in the second subsection but do have different system parameters as listed in Subsection 3.2.4.3.

#### **3.2.4.2.1 Common Workstation Parameters**

1. Worm Length Mean:  $X$  the value in bytes identified in the traffic analysis and modeling subsections, 3.1.3 and 3.1.4. The actual worm length is generated at random based on the distribution data provided in Appendix A.
2. DMA to NIC Delay: 44 MB/s the rate data flows from main memory to the network adapter per Myrinet's reported performance measurements<sup>6</sup>. However, this parameter can be used for any I/O bus as displayed in Chapter 4 where values corresponding to a PCI bus are used.
3. DMA from NIC Delay: 66 MB/s the rate data flows from the network adapter main memory to per Myrinet's reported performance measurements<sup>7</sup>. However, this parameter can be used for any I/O bus as displayed in Chapter 4 where values corresponding to a PCI bus are used.
4. My ID: *varying* an integer used to identify each workstation individually. This value is used in the final route flit for error detection by which an adapter can verify that a worm arrived at the correct destination.
5. Main Memory Size: 999 this value allows 999 different worms of varying length to be stored in "Main Memory" within a workstation. The storage is used when a worm is awaiting confirmation from the Myrinet Adapter before beginning the DMA delay. The value for this parameter may be increased to allow more worms to queue if needed or, to the value may be reduced if queuing is not probable. The smaller value would reduce system resource consumption during simulation.
6. MTU: 8KB the largest data size that the Myrinet TCP/IP stack can communicate within a worm.

#### **3.2.4.2.2 Simulated Traffic Workstation Parameters**

As previously stated, these parameters are associated with the workstations that are used in statistically simulated traffic patterns. The first ten are associated with the

---

<sup>6</sup> The data is provided at <http://www.myri.com/myrinet/SBus/m2f-sbus32.html>.

<sup>7</sup> Ibid.

“Controller” as described in Chapter 2. Conversely, parameters eleven to twenty are associated with the “Workers” detailed in Chapter 2.

1. Con Data Mean Delay Between Bursts: *1* the time unit in seconds that is used as a delay between *data* bursts within a workstation.
2. Con Mean Number of Pulses per Data Burst: *varying* the average number of *data* worms produced during a burst according to the simulated algorithm.
3. Con Mean Data Inter-Pulse Time: *varying* the average delay between *data* worm generation within a burst corresponding to the simulated algorithm.
4. Con Ave Data Size: *varying* the average *data* worm size in bytes related to the particular algorithm being investigated within a simulation.
5. Con Data Var: *varying* the variance for the average *data* worm size in related to the particular algorithm being investigated within a simulation.
6. Con Overhead Mean Delay Between Bursts: *1* the time unit in seconds that is used as a delay between *overhead* bursts within a workstation.
7. Con Mean Number of Pulses per Overhead Burst: *varying* the average number of *overhead* worms produced during a burst according to the simulated algorithm.
8. Con Mean Overhead Inter-Pulse Time: *varying* the average delay between *overhead* worm generation within a burst corresponding to the simulated algorithm.
9. Con Ave Overhead Size: *varying* the average *overhead* worm size in bytes related to the particular algorithm being investigated within a simulation.
10. Con Overhead Var: *varying* the variance for the average *overhead* worm size in related to the particular algorithm being investigated within a simulation.
11. Data Mean Delay Between Bursts: *1* the time unit in seconds that is used as a delay between *data* bursts within a workstation.
12. Mean Number of Pulses per Data Burst: *varying* the average number of *data* worms produced during a burst according to the simulated algorithm.
13. Mean Data Inter-Pulse Time: *varying* the average delay between *data* worm generation within a burst corresponding to the simulated algorithm.
14. Ave Data Size: *varying* the average *data* worm size in bytes related to the particular algorithm being investigated within a simulation.

15. Data Var: *varying* the variance for the average *data* worm size in related to the particular algorithm being investigated within a simulation.
16. Overhead Mean Delay Between Bursts: *1* the time unit in seconds that is used as a delay between *overhead* bursts within a workstation.
17. Mean Number of Pulses per Overhead Burst: *varying* the average number of *overhead* worms produced during a burst according to the simulated algorithm.
18. Mean Overhead Inter-Pulse Time: *varying* the average delay between *overhead* worm generation within a burst corresponding to the simulated algorithm.
19. Ave Overhead Size: *varying* the average *overhead* worm size in bytes related to the particular algorithm being investigated within a simulation.
20. Overhead Var: *varying* the variance for the average *overhead* worm size in related to the particular algorithm being investigated within a simulation

### 3.2.4.3 Simulation System Parameters

1. Total Number of Systems/Max Number of Workstations: *varying* this value corresponding to the number of workstations for a given simulation. The values used are 2, 4, 8, 16, 32, 64, or 128.
2. Workload Target: *varying* the target size in bytes for the simulation to complete. This value corresponds to the amount of data that must be communicated with the two different types of FFT algorithms emulated. Figure 3.2 shows the different values according to equation 3.4 for the Row Column version and equation 3.5 for the Vector Radix version.

2	786432	786432
4	1310720	1179648
8	1703936	1376256
16	2031616	1474560
32	2326528	1523712
64	2605056	1548288
128	2875392	1560576

Figure 3-2 Workload Targets in Bytes

$$RC = 8 * 3 * (p - 1) * (\text{image size} / p) \quad (3.4)$$

$$VR = 8 * (\text{image size} / p) * ((\log_2 p * p / 2) + 2 * (p-1)) \quad (3.5)$$

3. Message Header Size: 82B the value associated with the overhead that the Myrinet route and flow control symbols add to a worm.
4. Arrivals File: *varying* the named location of a file that correlates to the simulation time stamp for each worm. This parameter is only present when using the trace data as inputs.
5. Dest File to Open: *varying* the named location of a file that correlates to the destination stamp for each worm. This parameter is only present when using the trace data as inputs.
6. Length File to Open: *varying* the named location of a file that correlates to the length stamp for each worm. This parameter is only present when using the trace data as inputs.
7. Source Distribution String: *varying* the a list of workstation node identifiers used by the workstations to determine which node will create and initiate communications for each worm to be during a simulation. This parameter is only present when using the trace data as inputs.
8. Data Stats File to Open: *varying* the named location of a file that correlates to the saved statistics computer for each *data worm* communicated during a simulation. A data worm is defined in Chapter 2.
9. O/H Stats File to Open: *varying* the named location of a file that correlates to the saved statistics computer for each *overhead worm* communicated during a simulation. An overhead worm is defined in Chapter 2.
10. Number of Nodes: *varying* the total number of workstations and switches,  $n$ , which a given topology has during a simulation. This value maps to an  $n \times n$  matrix that is used for route generation. Appendix C provides an example of a routing matrix used during a simulation.
11. Image Size: 256 the dimensions of the  $n \times n$  matrix. This value is used by the *Workload Target* parameter to calculate the amount of traffic that must be simulated.
12. Link Length in Meters: 6 the length of the cables used to interconnect the nodes within the model. This value must correlate to the standards defined by the Myrinet specifications to allow the nodes to communicate at the 1.28Gbps speed.

13. Global Seed: *varying* the seed values used within the simulation where varying or random processes are used. These values were held constant from one simulation to the next to assist in the comparison of the different simulations as provided in Chapter 4.

### 3.2.5 *Measurements*

Typical computer models measure latency, the life span of a worm measured in a unit of time usually in seconds, and bandwidth, the amount of data communicated per unit of time usually in MB/s. This research effort compares each value in relation to the variations applied to a given topology. Each variation of a given topology will be compared in terms of bandwidth and in relation to one another using Jain's visual and *t-test* for unpaired systems as listed in Subsection 3.3.1.2.

### 3.2.6 *Simulations*

A given simulation is initialized at the topmost view by establishing the performance metrics and system-level memory variables. After initialization, the workstations begin generating worms based on the type of simulation, trace data or statistical representation, and on the algorithm load identified in Subsections 3.1.3 and 3.1.4. A data size is then generated and its corresponding latency is derived according to the type of simulation. The use of burst mode DMA transfer within a workstation correlates to transfer capability of 44MB/s going from main memory to the adapter and a rate of 66MB/s going from the adapter to main memory<sup>8</sup>. Actual worm inter-arrival time (the time between worm creation) is based on the loads identified in Subsections 3.1.3 and 3.1.4. After worm generation, a time stamp is entered into the worm's header data for latency measurement purposes. The newly created

---

<sup>8</sup> Different values may be substituted as in the case of the PCI tests listed in Figure 3-6.

worm has a destination generated based on the predetermined destination used within the trace data or on a randomly generated value in the case of the statistical representations with every node in the system having an equal probability.

After the destination is chosen, the source notifies the MCP of a pending worm. If the SRAM space is available, then the MCP DMA's the worm with a latency according to Equation 3.6. If the space is not available, then the worm sits in main memory in a first-in, first-out (FIFO) queue fashion until the MCP DMA's it.

$\frac{\text{Size of Worm in Bytes}}{\text{DMA Transfer Rate}} \quad (3.6)$
---

Following a DMA into the SRAM, a worm must wait in a FIFO fashion until the MCP selects it for transmission. Once a worm is chosen, the MCP initiates the route creation routine and inserts this data into the header effectively increasing the size of the worm. The MCP then initiates the transmission of the worm by generating and sending a GAP symbol. Per Myricom's specification, each node that receives a worm as an intermediate stop will consume a GAP symbol and generate a new one upon relaying the worm to its next hop. The worm's first flit is then sent to the next hop. Each flit then follows the same process as the first. If during transmission a STOP signal is received, the MCP ceases transmitting of flits and awaits a corresponding GO signal before transmission of a worm continues. When the MCP sends the final trailer flit, the SRAM is updated to reflect the available space.

If the first destination in the routing field is a switch, then certain steps must be performed prior to a worm continuing out through the appropriate output port. First, upon receiving a GAP on an input port, the port prepares to examine the next route flit to determine the requested output port. After determining the appropriate output port, the input port prepares the switch control signal to communicate its request to the arbitration logic. The arbitration then chooses which query to respond to and notifies the appropriate input port. If an output port is not available because another worm owns the port, the request is queued and the worm blocks in place at the input port. As subsequent flits arrive at the port, the flow control routine monitors the level of flits and will create necessary signals to stop the upstream flow. If the input port has received the output port, the arriving flits are gated through (incurring a delay identified in the *Average Path Latency* parameter). After the TRAILER symbol of the worm has been identified, the input port can begin processing the next arriving GAP symbol. In contrast, after the output port identifies the TRAILER symbol, it delays the worm to incur the SAN-to-LAN conversion latency. After the latency is incurred, the port is made available and the request queue is serviced in a round robin fashion.

After assigning an output port to an input port, the arbitration logic signals the output port to create and send a new GAP symbol. Also, after receiving notice that it has been assigned an output port, an input port will drop the first route flit but forward each successive flit whether it is route or standard data. The effect of dropping the first route flit at each hop is evident when the worm reaches its final destination. At the final destination, the remaining route flit should be equal to the receiving workstation's node id. This facilitates the error checking procedures within the model as each workstation can check an incoming worm to



verify that it arrived at its intended destination. In addition, the final route flit has a special one-bit flag that differentiates it from the other route flits. This allows the receiver to ensure that it was indeed the final destination for that worm.

Finally, the MCP acting as the receiving queue for a workstation also performs the flow control. In fact, each node performs the flow control on a link-by-link basis with only the nearest neighbor as its concern. Each node has an 80-byte buffer to receive flits. There is also a STOP and GO line (identified by the *Size of Region above the Stop Line* and *Size of Region below the Go Line* parameters) between which the a node can receive flits without delay. If the buffer reaches the STOP line, the node will send a STOP signal. Once the buffer has been emptied to at least the GO line, it will send the GO signal. Received worms at the final destination are buffered in the STOP/GO queue until the data size field is received. Upon receipt, the MCP checks the available SRAM space. If enough space is available to completely receive the worm, then the MCP 1) allocates the needed space, 2) copies the header fields into the allocated region, and 3) continues with the receiving process. Else, the flow control sends the STOP signal and polls the SRAM until it receives the necessary space. Figure 3-3 lists the actual simulations executed for this research effort.

### ***3.3 Verification and Validation***

Verification and validation are the tools for ensuring a simulation model is built correctly and it accurately defines the system under study [Jai91]. Both processes provide credibility to the model and substantiate the data obtained from the model. Verification asks whether the interacting components accurately represent the system in question [BaC96].

Validation questions whether the model's data is accurate through tests that compare the model's performance with the actual system's performance when available [BaC96].

### **3.3.1 Design Verification**

This subsection focuses on the "debugging" of the simulation model developed in this research effort. Jain lists eleven techniques for debugging a simulation model [Jai91]. His claim is that any of the set could be used in combination to adequately verify a model's operation. The techniques that Jain identifies that are applicable to this research are:

- Top-Down Modular Design
- Antibugging
- Structured Walk-Throughs
- Running Simplified Cases
- Degeneracy Tests
- Seed Independence

Verification for this research consisted of three phases. First, the model's construction was verified. Then, the tests conducted on the model were verified. And, finally the measurements taken were verified. In each case, the use of Designer's Module Block Verification routine verified that the graphical depiction of a block was syntactically correct. Furthermore, the use of Designer's Interactive Simulation Manager (ISM) allowed the model to be verified through an interactive process of step and test debugging. Whenever warnings or errors were encountered the ISM facilitated debugging by graphically displaying the state information.

Identifier	Traffic Type	Nodes	Algorithm
PCI_33	Trace	2	Row Column Basic
PCI_66	Trace	2	Row Column Basic
RCB_Trace_One_File	Trace	2	Row Column Basic
RCB_Trace_2	Trace	2	Row Column Basic
RCB_Stat_2	Stat	2	Row Column Basic
RCB_Trace_4	Trace	4	Row Column Basic
RCB_Stat_4	Stat	4	Row Column Basic
RCB_Stat_8	Stat	8	Row Column Basic
RCB_Stat_16	Stat	16	Row Column Basic
RCB_Stat_32	Stat	32	Row Column Basic
RCB_Stat_64	Stat	64	Row Column Basic
RCB_Stat_128	Stat	128	Row Column Basic
RC1_Trace_2	Trace	2	Row Column Optimization Num. 1
RC1_Stat_2	Stat	2	Row Column Optimization Num. 1
RC1_Trace_4	Trace	4	Row Column Optimization Num. 1
RC1_Stat_4	Stat	4	Row Column Optimization Num. 1
RC1_Stat_8	Stat	8	Row Column Optimization Num. 1
RC1_Stat_16	Stat	16	Row Column Optimization Num. 1
RC1_Stat_32	Stat	32	Row Column Optimization Num. 1
RC1_Stat_64	Stat	64	Row Column Optimization Num. 1
RC1_Stat_128	Stat	128	Row Column Optimization Num. 1
VRB_Trace_2	Trace	2	Vector Radix Basic
VRB_Stat_2	Stat	2	Vector Radix Basic
VRB_Trace_4	Trace	4	Vector Radix Basic

Identifier	Traffic Type	Nodes	Algorithm
VRB_Stat_4	Stat	4	Vector Radix Basic
VRB_Stat_8	Stat	8	Vector Radix Basic
VRB_Stat_16	Stat	16	Vector Radix Basic
VRB_Stat_32	Stat	32	Vector Radix Basic
VRB_Stat_64	Stat	64	Vector Radix Basic
VRB_Stat_128	Stat	128	Vector Radix Basic
VR2_Trace_2	Trace	2	Vector Radix Optimization Num. 2
VR2_Stat_2	Stat	2	Vector Radix Optimization Num. 2
VR2_Trace_4	Trace	4	Vector Radix Optimization Num. 2
VR2_Stat_4	Stat	4	Vector Radix Optimization Num. 2
VR2_Stat_8	Stat	8	Vector Radix Optimization Num. 2
VR2_Stat_16	Stat	16	Vector Radix Optimization Num. 2
VR2_Stat_32	Stat	32	Vector Radix Optimization Num. 2
VR2_Stat_64	Stat	64	Vector Radix Optimization Num. 2
VR2_Stat_128	Stat	128	Vector Radix Optimization Num. 2
VR4_Trace_2	Trace	2	Vector Radix Optimization Num. 4
VR4_Stat_2	Stat	2	Vector Radix Optimization Num. 4
VR4_Trace_4	Trace	4	Vector Radix Optimization Num. 4
VR4_Stat_4	Stat	4	Vector Radix Optimization Num. 4
VR4_Stat_8	Stat	8	Vector Radix Optimization Num. 4
VR4_Stat_16	Stat	16	Vector Radix Optimization Num. 4
VR4_Stat_32	Stat	32	Vector Radix Optimization Num. 4
VR4_Stat_64	Stat	64	Vector Radix Optimization Num. 4
VR4_Stat_128	Stat	128	Vector Radix Optimization Num. 4

Figure 3-3 Simulations

### 3.3.1.1 Construction

Following Jain's *Top-Down Modular Design* [Jai91] allowed the development phase to produce blocks, called dummy blocks, that were void of functionality but produced an output that provided the predetermined, correct result based on the input(s). For example, during the development of the top-level system view, the workstation block was placed without the necessary three processes (Source, Myrinet Adapter, and Destination). However,

for verification purposes, the workstation was given the necessary functionality to run in a single switch environment.

First, two workstations were connected to one another without an intervening switch. This allowed the top-down development of the workstations to verify that a system could generate a worm and transfer it to another system. This also verified that a system could adequately receive a worm. The original tests were conducted to avoid any queuing of the flits so flow control was not a concern. In addition, the size of the worms was smaller than the 80-byte flow control buffer. After verifying the initial construction, the functionality of the workstations was gradually increased. Following each enhancement, new tests were conducted that met the original goals and met the new requirements. After the workstations were verified functional for the small two-node case, eight workstations were connected to a switch with each workstation designed to send to each of the other workstations in an iterative fashion. The workstations produced a worm with a set destination and valid route for traversing the switch. The generated worms were smaller than the 80-byte flow control buffer and were adequately timed to avoid the need to provide flow control. Gradually, functionality was enhanced inside the switch via the top-down approach to implement the necessary features. Verification was accomplished within this simplistic design by placing probes at the input and output ports of the blocks to ensure that the correct operational paths were being followed. In addition, the output data was compared with the expected output.

After verifying the operation of the network in a single switch topology, the top-down design process continued for the construction of more advanced topologies. At this point, each component was worked on until its development was stalled due to a lack of

functionality in another component. The first component to be enhanced was the switch. After each major sub-component was designed, the switch was tested to ensure that the correct results were achieved. The testing monitored the arbitration to ensure that the round robin assignment proceeded fairly, that the appropriate delays across the switch were incurred, and that the correct output port handled the requests while the incorrect output ports ignored it. Full verification of the switch was dependent on the implementation of the workstation (tests of the actual contention for an output port and the actual node-node communication were dependent).

The final component to be implemented was the workstation. The workstation's development followed the top-down design process as the Source process, the Myrinet Adapter, and then the Destination process were enhanced in order. Again, verification tests were conducted following each phase. After designing the Source, the Workstation was tested to ensure that the proper inter-arrival time was met according to the appropriate load and that the valid network node received the generated worm. The next step was to create the Myrinet Adapter process. Following the design of the process, the Workstation was tested again to ensure that 1) the correct routing data was used, 2) the adapter used the SRAM as designed to include the appropriate delays, and 3) that the adapter sent and received the appropriate hand-shaking signals. Tests of the flow control were delayed pending development of the final process, the Destination block. The Destination block is responsible for processing the latency statistics for a given worm. Testing of this block focused on ensuring the correct statistics were computed for each worm. To operate correctly, the block was required to accept the received worms. The Adapter process was also tested for injection of flow control signals at the appropriate time, to use the SRAM as

designed, and to free resources upon receipt of the trailer flit. Each of these areas was verified to be operating correctly. In addition, the latency calculations were verified within the ISM.

The final class of verification tests centered on ensuring correct routing. Each of the seven topologies was tested to verify each node could successfully communicate with every other node. In addition, these tests verified that the routing table was correct and that the topology layout was correct. Additional tests verified the workstation and the switch under loads of 75%, 100%, and 125%. The workstations were monitored to ensure that the worms were handled in sequence and that the Myrinet Adapter functioned as expected. The switches were monitored to ensure that fair, round robin arbitration occurred and that the switches connected input- to output-ports together per the arbitration. Also, the switches were monitored to ensure that worms were routed to the correct output-port and that the input port held new worms until the requested output-port was assigned.

### ***3.3.1.2 Measurements***

Subsection 3.2.5 introduced the metrics of interest for this effort. In addition to the stated metrics, the mean, standard deviation, and the confidence level for the latency and bandwidth are computed. The steps to verify the measurements were:

1. Conduct preliminary tests for each configuration. These tests were executed until steady state was first reached using different random number seeds. In addition, metrics were tracked that provided 1) the wall clock run time and 2) the simulation time to reach steady state.

2. Obtain the number of independent replications. These values represented the corresponding 90%, 95%, and 99% confidence intervals with accuracy of one percent as computed for the bandwidth of each configuration. Refer to Appendix A for this data.
3. Determine the appropriate confidence interval and run simulations. Based on the run times obtained in bullet numbers one and two this effort ran simulations that targeted 90% confidence intervals. This value was chosen because it best represented the bandwidths of the data worms and the bandwidths of the overhead worms. In addition, each simulation was executed for a total simulation time interval that exceeded eight times the minimum steady-state time value [BaC96].
4. Determine the average data bandwidth across each independent replication. Compute the sample mean, sample variance, and the confidence interval for the mean.
5. Obtain the quantile of  $t$  distribution [Jai91] and extrapolate the confidence level from the  $t$  distribution table according to Equation 3.7 where  $r$  is the desired accuracy in  $\pm r\%$ ,  $\bar{x}$  is the sample mean,  $\sigma$  is the sample standard deviation, and  $m$  is the number of replications.

$$t \equiv \frac{\bar{x}r\sqrt{m}}{100\sigma} \quad (3.7)$$

6. Plot the confidence intervals from each algorithm for a given topology (provided in Chapter 4). Conduct an approximate visual test to determine if one algorithm is better than the others, if not, produce the  $t$ -test for unpaired systems [Jai91].

Preliminary tests were conducted to achieve a sample mean for each configuration's latency and bandwidth resulting from the initial steady-state interval. The obtained data is displayed in Appendix A. Jain's formula for determining sample size is used to support the confidence intervals and is displayed in Equation 3.8 where  $r$  is the desired confidence factor,  $t$  is the student- $t$  distribution for the respective confidence interval,  $s$  is the sample variance, and  $\bar{x}$  is the sample mean. Since latency and bandwidth are directly correlated to one another [Jai91], this effort determined the number of replications based on the bandwidth calculations. In an instance where  $n$  evaluated to a number less than Jain's recommended ten repeated trials, then this effort used  $m$  equal to ten. The use of a student- $t$  distribution is

justified, as the number of replications is less than thirty. In Chapter 4, an analysis is conducted to contrast the performance of the different algorithms over a given topology as supported by Jain's approximate visual test [Jai91]. Jain states that a visual test is conclusive if the confidence intervals (CIs) do not overlap and one alternative is said to be superior to the other. If the CIs overlap and the mean of one is in the CI of the other, then the alternatives are not different. However, if the CIs overlap but the mean of any one is not in the CI of the other, then a *t-test* for unpaired systems would need to be performed [Jai91]. Chapter 4 discusses the comparisons while Appendix A contains all of the simulations' data according to the measurement criteria listed in this subsection.

$$n \geq \left( \frac{100ts}{r\bar{x}} \right) \quad (3.8)$$

### 3.3.1.3 Simulations

The system configurations contained in this research provided a lot of room for errors. Considering the number of blocks in the 128-node configurations, it was very probable to encounter an error in semantics such as assigning the wrong node id to a workstation. Simulations were verified for operational correctness by looking at each test case individually for 1) freedom from block errors, 2) correct assignment of parameters, and 3) complete delivery of worms. The first area of concern was met through Designer's verification "compilation" where omission of parameter assignments, omission of port connections, and lack of syntax adherence are checked. 1) Designer's ISM and 2) peer



review verified the second and third areas of concern. The ISM allowed the simulations to be checked while running an actual test. Peer review verified the correct assignment of parameters and the correct delivery of worms. Also, the Myrinet Adapter block was designed to signal an error condition whenever a worm reached it in error (occurs if a worm has bad routing data).

### **3.3.2 *Model Validation***

The validation process differs from the verification process by targeting the actual results viewed in an operational system. Jain notes that actual validation is model dependent but all paths seek a common goal of justifying the assumptions to meet actual system results [Jai91]. Jain defines three keys to validation:

1. Assumptions
2. Input parameters
3. Output results

In addition, Jain recommends collecting data through validity tests via:

1. Expert Intuition
2. Real System Measurements
3. Theoretical Results

As noted in [Jai91], the lack of resources is often the reason that simulations are undertaken. And, this effort is no exception. Due to the limited resources available, Jain recommends validation of simplistic models through available means. Thus, the following configurations were used during validation tests:

1. Two nodes with one interconnecting link and no switch. This allows the perfect one-way traffic environment where no interfering process can contend for system resources.
2. Two nodes with a single switch and two interconnecting links. This allows the latency incurred by a worm traversing a switch to be approximated.

#### ***3.3.2.1 Parameters and Assumptions Validation***

Expert intuition is termed the “most practical and common way” to conduct validation of a model [Jai91]. In tune with this statement, this effort sought the advice of the Myricom, the manufacturer of Myrinet. Various aspects of the design were presented to the technical staff for feedback. Their support was invaluable to the success of this project. Specific areas of support included validation of the Switch design and of the Myrinet Adapter design. Whereas the first generation Myrinet Switch is well documented, the second generation’s documentation is still pending. Thus, expert consultation was used extensively to validate the parameters and assumptions used in the modeling of the switch. The Myrinet network interface is documented and some simplifying assumptions were made to alleviate unnecessary overhead that leads to increased run times. These assumptions were validated through Myricom’s support. The most important simplifying assumption was centered on the elimination of the network mapping and the overhead associated with it. Myricom estimates that 6% of the network traffic can be classified as overhead associated with maintaining the network map.

#### ***3.3.2.2 System Validation***

The validation of the system configurations was partially met through real-system measurements as detailed at the beginning of this section. The obvious exception to this centers on the topologies that use more than one switch. The network topology containing

sixteen or more workstations was designed based on a similar topology identified in [MaS97]. In addition, sensitivity tests were conducted to verify the congestion within the network. Increasing the average data size of the worms resulted in improved throughput while decreasing the average resulted in reduced throughput.

### 3.3.2.3 Results Validation

Figure 3-4 shows the first validation test. The results achieved in the simplistic test simulations were compared with the actual system results and with theoretical values. The first simulation test measured the bandwidth a worm achieves in the first validation test configuration while neglecting the overhead that results from the MPI environment. The data is then compared with the theoretical maximum bandwidth to ensure that the simulation values do not exceed them.

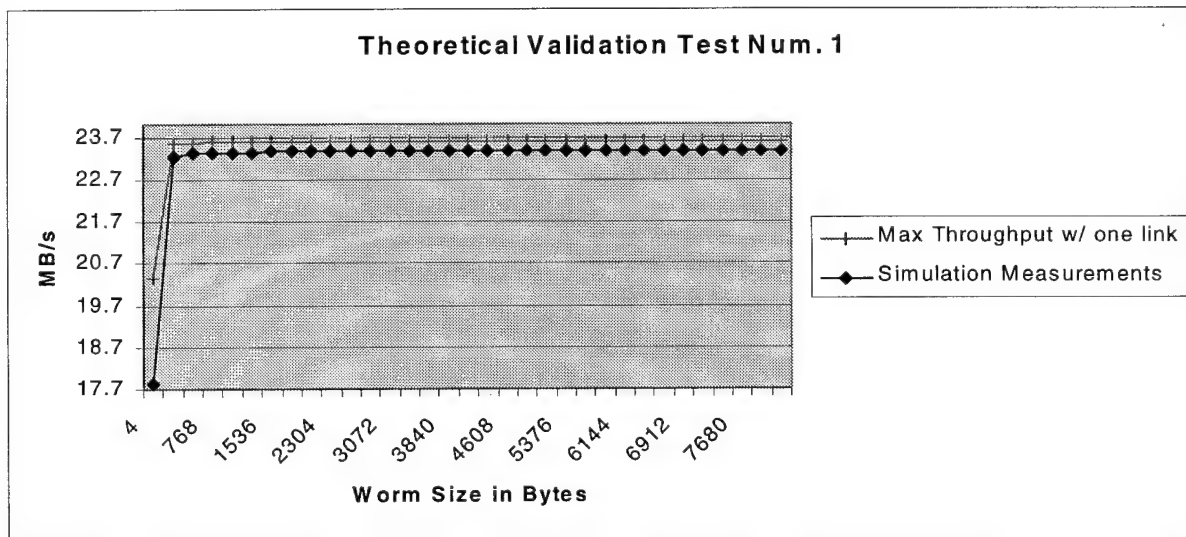


Figure 3-4 Theoretical Validation Test Number 1 Results

The second configuration was then tested with the simulated bandwidths compared to the theoretical maximums. Figure 3-5 shows the results obtained from this test. The third validation test compared the bandwidth achieved using the MPI overhead for configuration number one. The values obtained are plotted in Figure 3-6. The resulting average difference in bandwidths shows that the model achieves the desired bandwidth within .031%. The corresponding values obtained for configuration number two are plotted in Figure 3-7. The average difference obtained by the model is .045%.

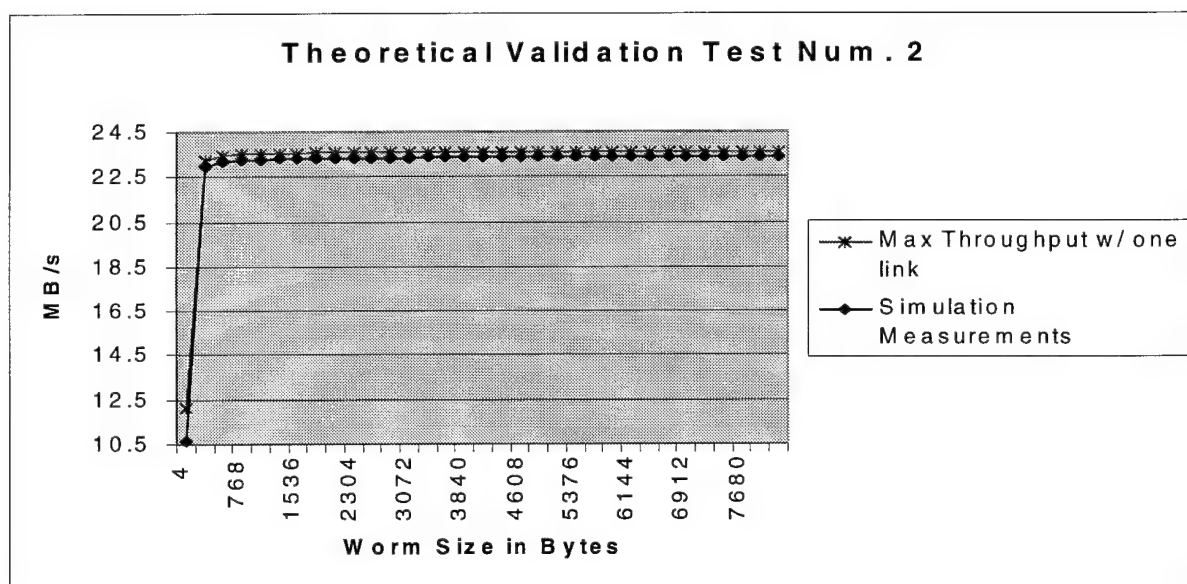


Figure 3-5 Theoretical Validation Test Number 2 Results

### 3.4 Summary

The methodology presented in this chapter can be used to analyze the performance of routing and connectivity support for DSIP applications in a Myrinet. The routing and connectivity data were supported through a simulation model using actual traffic analysis of FFT algorithms in a Myrinet. In addition, a general parallel processing environment's traffic

was introduced to further support message passing multi-computer implementations. The parameters used were sufficiently tested to ensure fairness in data collection. The top-down design through Designer allowed efficient verification and validation.

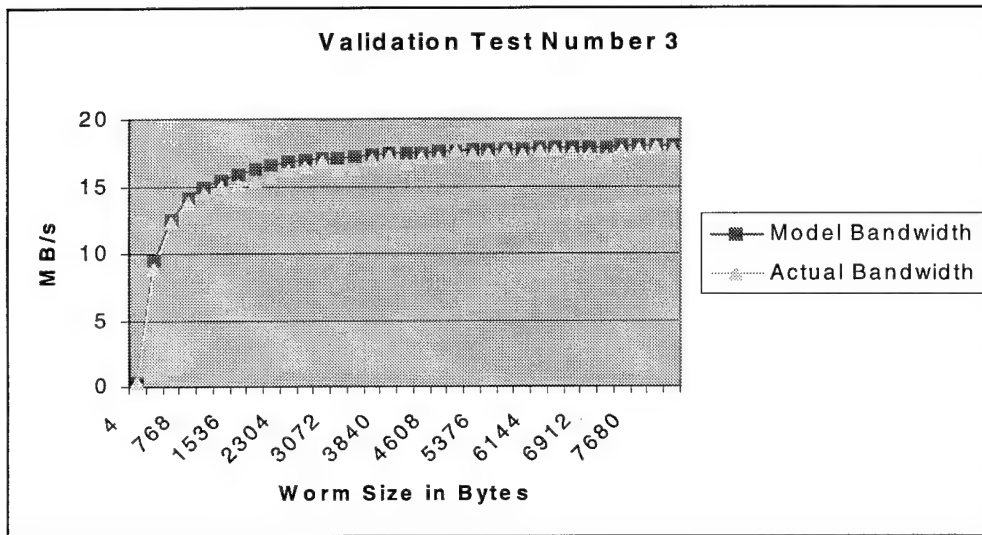


Figure 3-6 Validation Test Number 3 for Actual Measured Data vs. Simulated Data

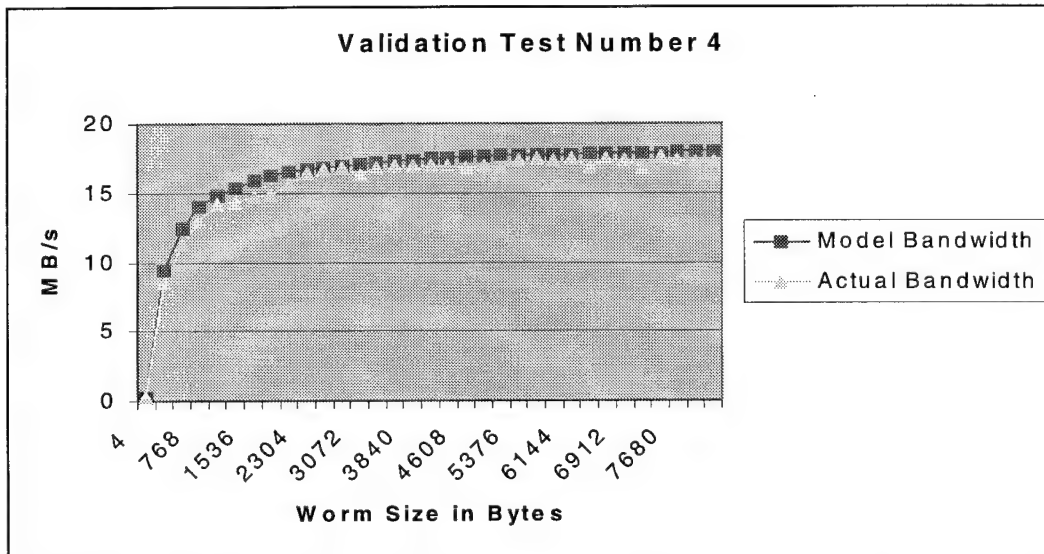


Figure 3-7 Validation Test Number 4 for Actual Measured Data vs. Simulated Data

## *IV. Analysis*

### *4.1 Introduction*

The data generated from the simulations outlined in Chapter 3 is analyzed in the following sections. Section 4.2 discusses the traffic trace analysis and the actual time and resources expended on the simulations. The data recorded for the trace simulations is analyzed in Section 4.3 while Section 4.4 compares the performance of each simulation to identify the algorithm that best utilized the Myrinet as the number of nodes was scaled. A summary of the chapter is provided in Section 4.5.

### *4.2 Traffic and Simulation Run Time Analysis*

As outlined in Chapter 3, a non-intrusive software trap was used to capture the network traffic across a Myrinet during the execution of the FFT algorithms under investigation. These captures identified characteristics about the traffic signature that were used to feed statistical engines that produced the desired traffic flow for higher order topologies. The reason for using a statistical engine was that resources were not available to configure topologies larger than the four-node configuration. The traces showed different patterns for both of the FFT algorithms as well as each variation of the two. The data obtained from the capture was feed to a program that was designed to calculate:

- the average size of the worm
- the variance of the data size
- the average time between packets within a given window of interest

- the statistical confidence to support a 99% confidence interval with an accuracy of one percent

The program was fed over 10,000 worms from each of the five variations of FFT algorithms. The results showed that a window size, or burst window, of one second provided the best scale to support the comparisons of the simulations against one another. Figure 4.1 shows the data obtained from the *RCB\_256\_2* and *RCB\_256\_4* captures. The corresponding scaled inputs used for this algorithm, *Row Column Basic*, and the complete data for the other algorithms is provided in Appendix A.

( a ) Con Data				( b ) Con Overhead			
	RCB_256_2	RCB_256_4	%+/-		RCB_256_2	RCB_256_4	%+/-
Burst Inter-arrival Time	1.00	1.00	0.00%	Burst Inter-arrival Time	1.00	1.00	0.00%
Ave. Number in Burst	24.50	10.50	-57.14%	Ave. Number in Burst	26.52	17.61	-33.57%
Ave. Delay Btwn Pkts	0.04	0.08	108.98%	Ave. Delay Btwn Pkts	0.09	0.19	119.46%
Size Variance	35004596.75	7444840.55	-78.73%	Size Variance	72648.56	140659.30	93.62%
Ave. Size	8134.76	8101.16	-0.41%	Ave. Size	70.08	89.10	27.13%
( c ) Worker Data				( d ) Worker Overhead			
	RCB_256_2	RCB_256_4	%+/-		RCB_256_2	RCB_256_4	%+/-
Burst Inter-arrival Time	1.000	1.000	0.00%	Burst Inter-arrival Time	1.000	1.000	0.00%
Ave. Number in Burst	8.500	9.595	12.89%	Ave. Number in Burst	33.769	19.654	-41.80%
Ave. Delay Btwn Pkts	0.003	0.007	152.66%	Ave. Delay Btwn Pkts	0.054	0.174	222.00%
Size Variance	8537024.221	17942406.646	110.17%	Size Variance	59796.380	81019.359	35.49%
Ave. Size	8069.529	8211.224	0.018	Ave. Size	61.551	64.555	4.88%

Figure 4-1 Row Column Basic Algorithm Trace Data

In addition to feeding the simulations, the captured data was analyzed to determine the appropriateness of using the “bursty” traffic generators. As discussed in Chapter 3, the variance-time plot method [LeT94] was used to establish the validity of this generator. And

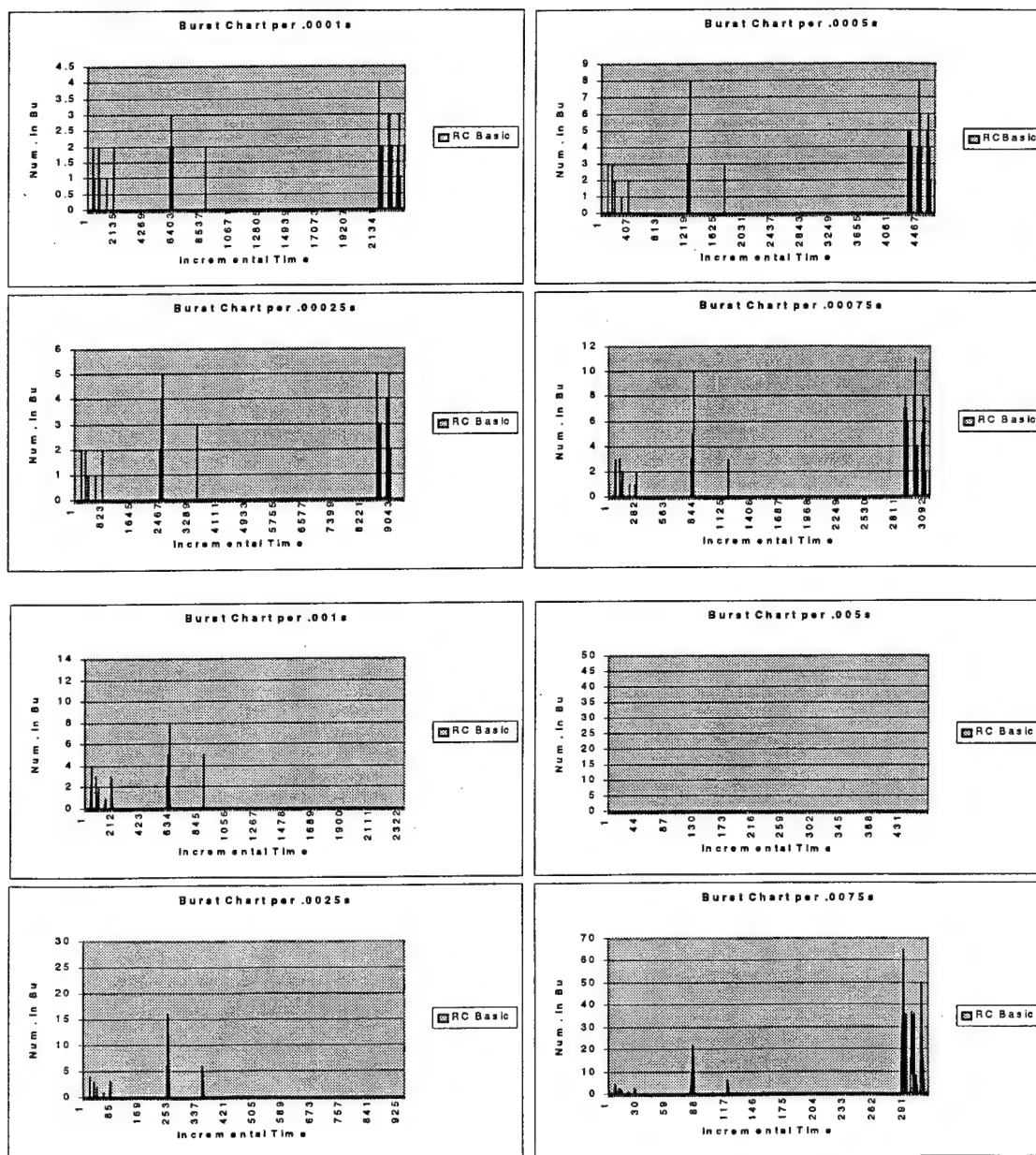


Figure 4-2 Packet Count v. Time for the Row Column Basic Algorithm for .0001 to .0075s



as Leland, et al, note, the visual method for obtaining self-similar patterns provides strong evidence that the algorithms under investigation indeed follow a strong bursty pattern. Using the time scales discussed in Chapter 3, Figures 4-2 and 4-3 show the plots for the Row Column Basic algorithm.

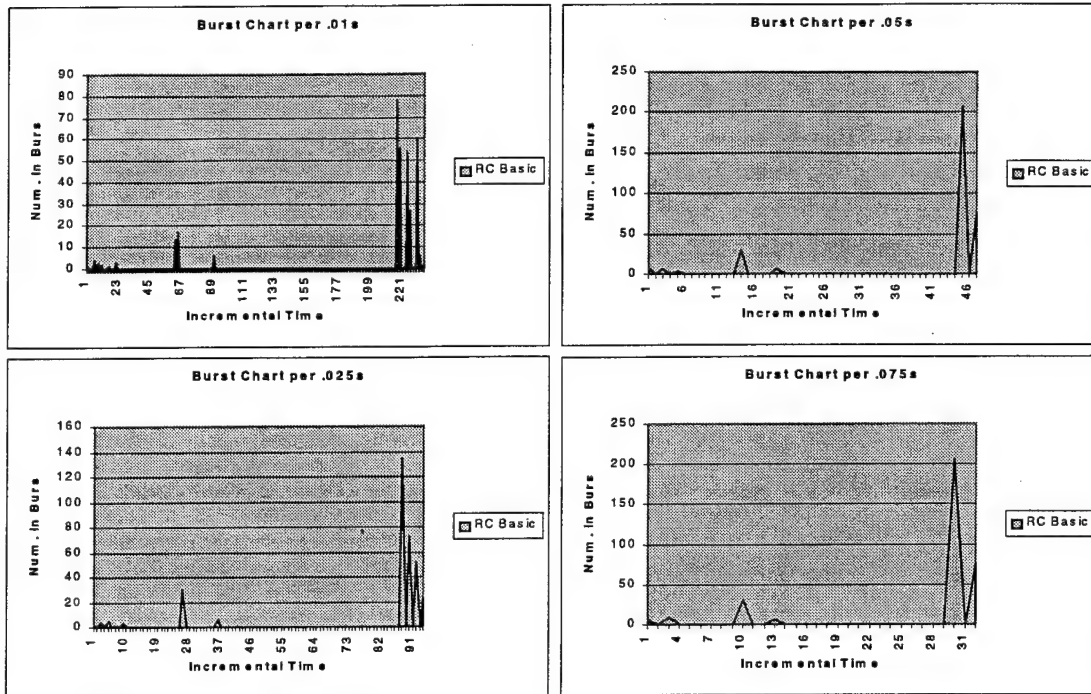


Figure 4-3 Packet Count v. Time for the Row Column Basic Algorithm for .01 to .075s Intervals

The strikingly similar patterns led to the decision to use the bursty method of generating the traffic and the corresponding scaled values as discussed earlier. The remaining algorithms are plotted in Appendix A. But, as important as the traffic patterns were to the performance analysis of the simulations, the analysis of the simulation execution run-time was equally important to the success of this research effort. If inappropriate parameters are used, then the execution time of the simulations could easily exceed the time schedule allotted for them. Therefore, initial tests were conducted to determine the

approximate run time for varying sizes of the matrix image for the topologies under investigation. As a result, it was determined that the 256 x 256 matrix would best fit the time schedule available to run the simulations. Figure 4-4 shows the resulting calendar days used for the different simulations. In each case, the simulations were performed with ten different seed values to improve the statistical confidence of the results. Also, a network of twenty-five Sun SPARCstation 20 machines served as the computing platform for the execution of the simulations.

#### ***4.3 Trace Simulation Performance Analysis***

The ability to accurately portray the system dynamics is paramount to the success of a computer model. Using a method such as Jain's validity tests [Jai91] serves to ensure that a reasonable facsimile of the system under investigation is produced. Therefore, as outlined in Chapter 3, several tests were run that showed that the model used in this investigation achieved similar performance to the real system measurements in each of the three validation tests outlined in Section 3.3.2. In fact, the simulations all exceeded a 95% confidence interval with an accuracy of one percent. In other words, statistical confidence in the model shows that the performance closely approximates that achieved in a real system. In fact, the validation tests each had less than a .01 percent difference in the performance results.

Simulation Identifier	Traffic Type	Topology	Execution Time
PCI_33	Trace	2	0.5
PCI_66	Trace	2	0.5
RCB_Trace_One_File	Trace	2	0.5
RCB_Trace_2	Trace	2	0.5
RCB_Stat_2	Stat	2	0.5
RCB_Trace_4	Trace	4	0.75
RCB_Stat_4	Stat	4	0.75
RCB_Stat_8	Stat	8	1.2
RCB_Stat_16	Stat	16	2.4
RCB_Stat_32	Stat	32	5
RCB_Stat_64	Stat	64	7.8
RCB_Stat_128	Stat	128	10.3
RC1_Trace_2	Trace	2	0.5
RC1_Stat_2	Stat	2	0.5
RC1_Trace_4	Trace	4	0.75
RC1_Stat_4	Stat	4	0.75
RC1_Stat_8	Stat	8	1.2
RC1_Stat_16	Stat	16	2.4
RC1_Stat_32	Stat	32	5
RC1_Stat_64	Stat	64	7.8
RC1_Stat_128	Stat	128	10.3
VRB_Trace_2	Trace	2	0.5
VRB_Stat_2	Stat	2	0.5
VRB_Trace_4	Trace	4	0.75
VRB_Stat_4	Stat	4	0.75
VRB_Stat_8	Stat	8	1.4
VRB_Stat_16	Stat	16	3
VRB_Stat_32	Stat	32	6.2
VRB_Stat_64	Stat	64	8.9
VRB_Stat_128	Stat	128	13.2
VR2_Trace_2	Trace	2	0.5
VR2_Stat_2	Stat	2	0.5
VR2_Trace_4	Trace	4	0.75
VR2_Stat_4	Stat	4	0.75
VR2_Stat_8	Stat	8	1.4
VR2_Stat_16	Stat	16	3
VR2_Stat_32	Stat	32	6.2
VR2_Stat_64	Stat	64	8.9
VR2_Stat_128	Stat	128	13.2
VR4_Trace_2	Trace	2	0.5
VR4_Stat_2	Stat	2	0.5
VR4_Trace_4	Trace	4	0.75
VR4_Stat_4	Stat	4	0.75
VR4_Stat_8	Stat	8	1.3
VR4_Stat_16	Stat	16	2.8
VR4_Stat_32	Stat	32	5.9
VR4_Stat_64	Stat	64	8.1
VR4_Stat_128	Stat	128	11.8

Figure 4-4 Simulation Execution Time in Calendar Days

With the model validated, the first series of simulations executed were those using the captured data as input. Figure 4-5 shows the performance data with the 90% confidence intervals printed for each of the 2-node tests. The plot shows that the Row Column Optimization 1, the Vector Radix Base, and the Vector Radix Optimization 4 tests achieved average bandwidths within 4.2817E-4% of one another. Thus, according to Jain [Jai91], since the confidence intervals overlap and the mean of each is in the CI of the others; each is statistically equivalent at the 90% confidence interval. Figure 4-6 shows the tests plotted together with a better view of the confidence intervals. In Figure 4-7, the data for the 4-node tests is displayed.

For this class of tests, the algorithms all achieve very similar bandwidth numbers. Again, the confidence intervals overlap and the mean of each is in the CI of the others thus, each is statistically equivalent at the 90% confidence interval. In addition to the two test classes, three tests were conducted using the captured Row Column Basic 2-node algorithm's traffic. The first test was conducted to display the effects of the MPI overhead on the overall performance calculations for a given simulation. As discussed in Chapter 2, the algorithms that perform the FFT computations use the MPI standard for programming in a parallel-processing environment. Underlying within the standard are different types of communication commands that can be utilized.

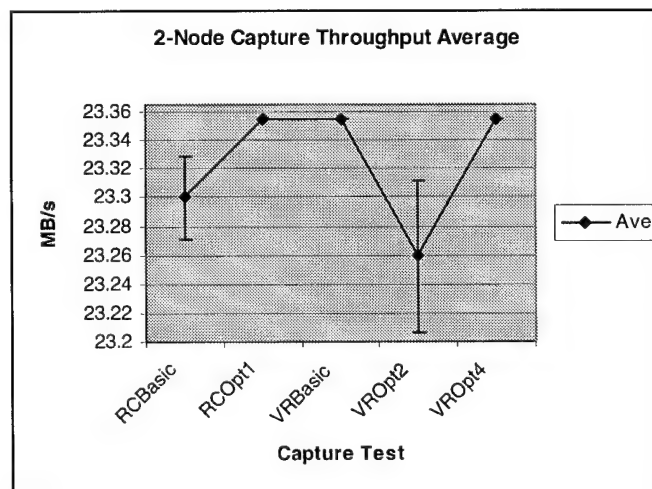


Figure 4-5 Data Message Throughput for 2-Node Capture Simulations

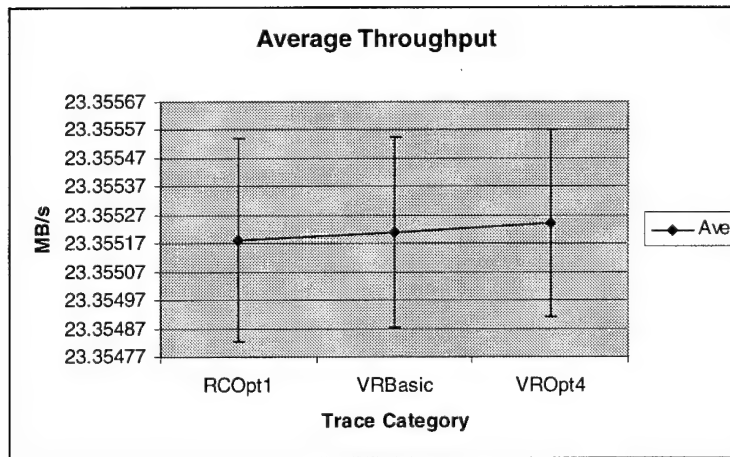


Figure 4-6 Average Throughput at 90% Confidence Interval for 2-Node Simulations

Furthermore, the data communications initiated by the FFT algorithms showed a distinct process that MPI employed for the commands used by the programs. That is, when one of the programs initiated, there was a certain amount of overhead that was performed to setup each of the workstations in the environment. Following the setup, the systems began to initiate the data communications. Accompanying each data worm was at least one worm that

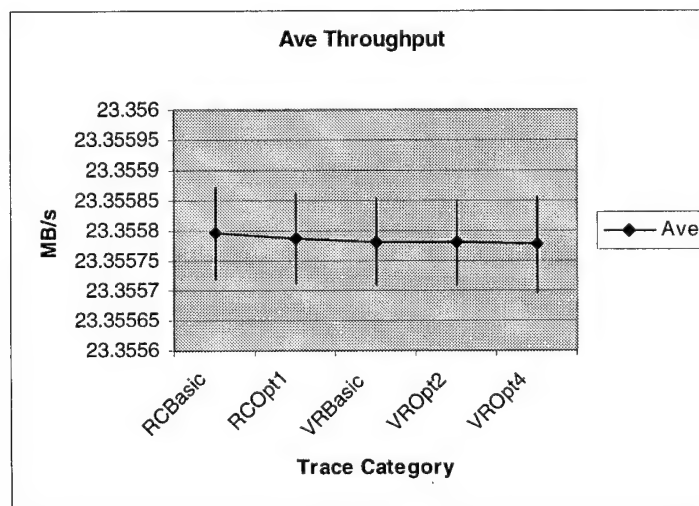


Figure 4-7 Data Message Throughput for the 4-Node Capture Simulations

acted as an acknowledgement request for the receiver. In turn, the receiver would send another worm acknowledging the sender's request. Finally, the message-passing environment created worms to handle the closing of the communicating processes for each node active in the parallel computations. The biggest problem with these overhead worms was that each was generally small in size. And, as displayed in Chapter 2, these small messages were low in bandwidth, which resulted in a combined performance of 21 MB/s, over 2 MB/s less than the bandwidth achieved by the data worms.

The next two tests performed used the *RCB\_256\_2* data to show the performance that the data messages could achieve had a PCI based system been simulated. The differences in the parameters fed to the model followed Myricom's data<sup>9</sup>. First, a 33 MHz class machine was used for each of the workstations. The differences between the PCI and the SBus systems required the PCI adapter to operate at a speed of 15ns as opposed to the SBus's 6.25ns. However, the DMA to NIC rate improved dramatically from 44 MB/s to 126.9 MB/s. Similarly, the DMA from NIC rate improved from 66 MB/s to 126.8 MB/s. The net result showed the PCI 33 MHz configuration achieving a bandwidth performance of 33.26 MB/s, which is nearly a 10 MB/s improvement over the SBus version. The final test case increased the PCI's bus rate to 66 MHz, which resulted in a speed improvement for the PCI's adapter that operated at 7.5ns. The data bandwidth further improved to 44.32 MB/s.

---

<sup>9</sup> Obtained at <http://www.myri.com>.

## **4.4 Comparative Analysis**

The comparative analysis conducted was based on the data collected from the “Stat” test cases. Recalling from Chapter 3, the “Stat” tests were executed using the traffic analysis conducted on the various algorithms to statistically model the traffic generated by each algorithm. The “bursty” traffic was emulated using Designer’s *Bursty Pulse Train*. Each of the five FFT algorithm variations was compared for each of the seven topologies. Subsections 4.4.1 through 4.4.7 discuss the results based on the topology. Subsection 4.4.8 provides a discussion of the results based on an aggregation of each FFT algorithm to summarize performance across all of the topologies and, Subsection 4.4.9 summarizes the results.

### **4.4.1 2-Node Topology Comparisons**

The data collected for the FFT algorithms under the 2-node category resulted in a statistical draw. As the 90% confidence interval (CI) plot shows in Figure 4-8, the five alternatives cannot be statistically differentiated. Each interval overlaps and the mean of each is contained within each of the confidence intervals. Therefore, no conclusion can be drawn as to which algorithm would be preferred for this topology using this statistical representation. However, since the trace data is available to support this topology, Figure 4-6 shows that three of the algorithms can be differentiated at this CI but not conclusively and further testing on actual systems is needed.

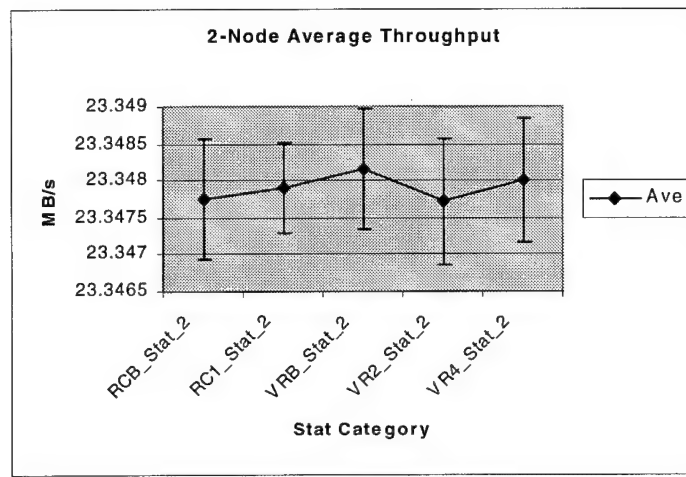


Figure 4-8 2-Node Throughput for Stat Tests

#### 4.4.2 4-Node Topology Comparisons

Contrary to the conclusion drawn about the 2-node tests, the 4-node tests showed that the *VR4\_Stat\_4* result was statistically better at the 90% CI than the other four. Figure 4-9 shows the 90% CI plots for the five tests that clearly shows the one algorithm achieving better performance than the others. The remaining four alternatives can not be statistically differentiated. Their intervals overlap and their means are contained within each of the other confidence intervals. Figure 4-7 showed that the “capture” tests resulted in a statistical draw for the algorithms.

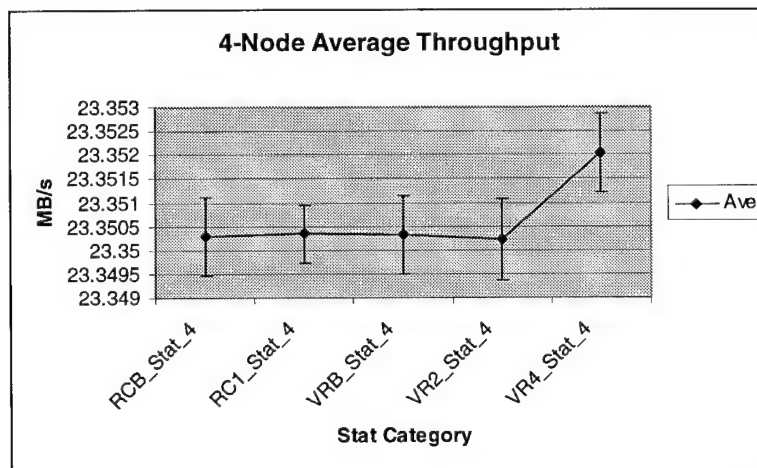


Figure 4-9 4-Node Throughput for Stat Tests



#### 4.4.3 8-Node Topology Comparisons

The 8-node tests provided no clear statistical winner. However, as Figure 4-10 shows, the *RCB\_Stat\_8* test resulted in a throughput that was not statistically equivalent to the other four at the 90% CI. The remaining four alternatives cannot be statistically differentiated. Their intervals overlap and their means are contained within each of the other confidence intervals.

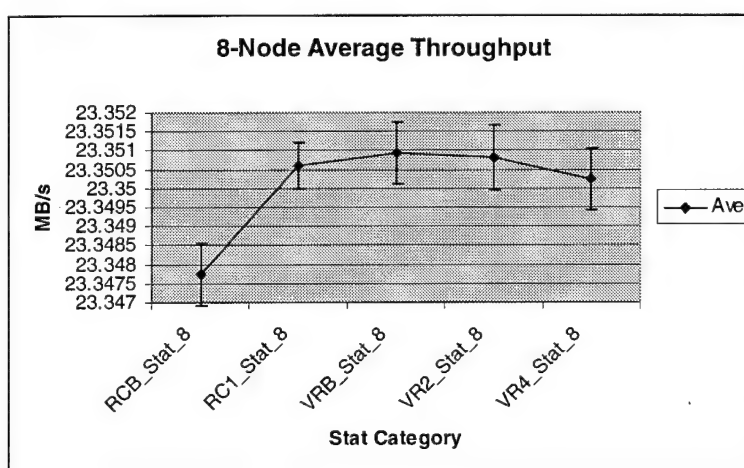


Figure 4-10 8-Node Throughput for Stat Tests

#### 4.4.4 16-Node Topology Comparisons

Figure 4-11 shows the 90% CI plots for the 16-Node “Stat” tests. Within this class of tests, two tests- *VRB\_Stat\_16* and *VR2\_Stat\_16*- achieved statistically equivalent and better performance than the other tests. Because the two intervals overlap and their means are contained within each of the other confidence intervals, the conclusion is drawn that the alternatives are not different. Of the remaining three, the *RCB\_Stat\_16* performance data is statistically inferior to the others and is considered the least desirable of the algorithms.

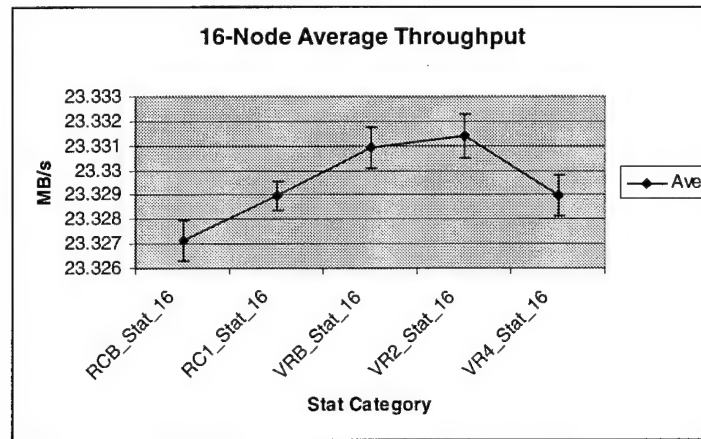


Figure 4-11 16-Node Throughput for Stat Tests

#### 4.4.5 32-Node Topology Comparisons

The data collected for the FFT algorithms under the 32-node category resulted in *RC1\_Stat\_32* and *VR2\_Stat\_32* being statistical equivalent. However, the 90% confidence interval (CI) plot shown in Figure 4-12(a) does not readily show this. If the plot were not conclusive, then a *t-test* would need to be performed to determine if one is superior to the other. As Subsection 3.3.1.2 stated, such an analysis is performed if the two tests in question have CIs that overlap but their means appear to be out of each other's CI. Before conducting the *t-test*, a second plot was produced to give a finer scale upon which to compare the two tests. The resulting graph, shown in Figure 4-12(b), allowed the conclusion to be made that the two tests were indeed statistically equal. As shown, each interval overlaps and the mean of each is contained within each of the confidence intervals. As was the case in the 8- and 16-node tests, the Row Column Basic Algorithm- *RCB\_Stat\_32*- statistically performed below the other alternatives.

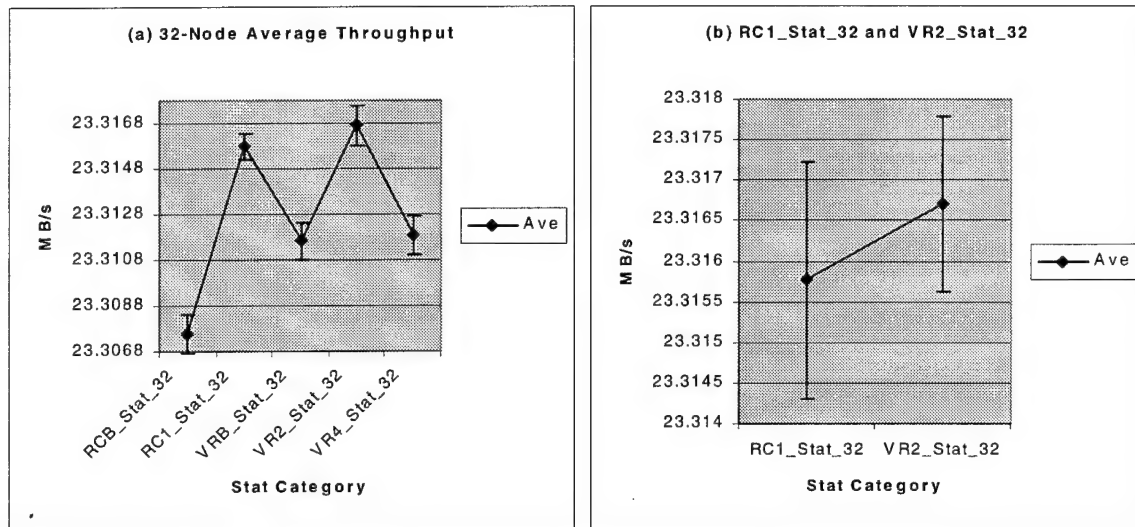


Figure 4-12 32-Node Throughput for Stat Tests

#### 4.4.6 64-Node Topology Comparisons

The 64-node tests showed that the *VRB\_Stat\_64* result was statistically better at the 90% CI than the other four. Figure 4-13(a) shows the 90% CI plots for the five tests that clearly shows the one algorithm achieving better performance than the others. However, it is not clear whether *RC1\_Stat\_64* or *VR4\_Stat\_64* are statistically equivalent. Subsequently, a second plot was produced to give a finer scale upon which to compare the two tests. The resulting graph, shown in Figure 4-13(b), allowed the conclusion to be made that the two tests were indeed statistically equal. As shown, each interval overlaps and the mean of each is contained within each of the confidence intervals. For this topology class, the Vector Radix Optimization 2 Algorithm- *VR2\_Stat\_64*- statistically performed below the other alternatives.

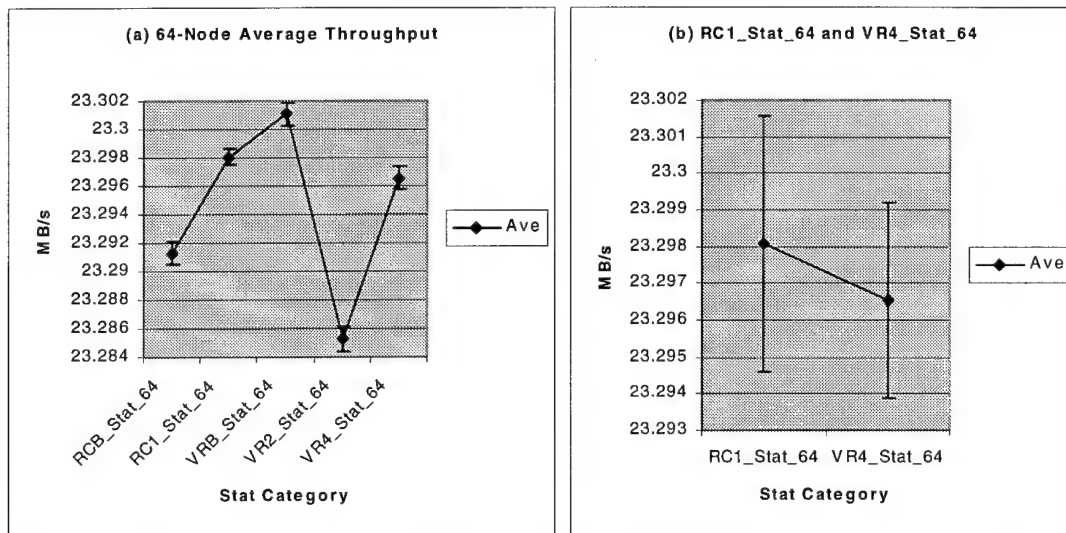


Figure 4-13 64-Node Throughput for Stat Tests

#### 4.4.7 128-Node Topology Comparisons

Figure 4-14(a) shows the 90% CI plots for the 128-Node “Stat” tests. The tests showed that the *RC1\_Stat\_128* result was statistically better at the 90% CI than the other four. As seen in Subsection 4.4.6, the graph does not easily allow differentiation among three of the remaining four plots. Figure 4-14(b) shows that the conclusion can be made that the three tests- *VRB\_Stat\_128*, *VR1\_Stat\_128*, and *VR4\_Stat\_128*- are indeed statistically equal. Once again, one test- *RCB\_Stat\_128*- is shown to be statistically inferior to the other four.

#### 4.4.8 Combined Topology Comparisons

The previous seven subsections displayed the statistical comparisons for each of the topology variations researched. In this subsection, each of the algorithms is compared as an aggregate of the performance achieved across all of the topologies. As a contrast, a score sheet is given that rates each of the algorithms based on their statistical topology performance.

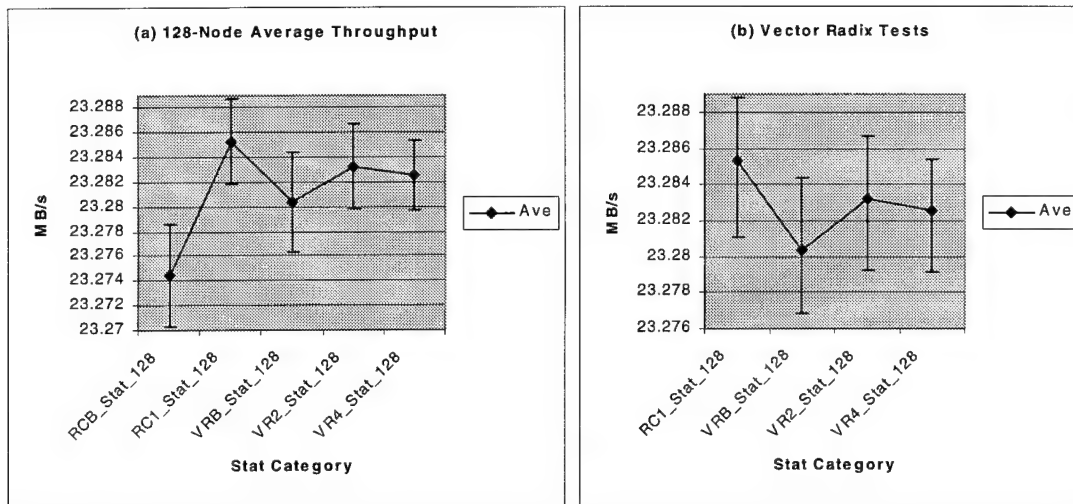


Figure 4-14 128-Node Throughput for Stat Tests

Similar to the statistical data previously shown, Figure 4-15(a) shows the combined performance achieved across all of the topologies. The visual plot analysis shows that at the 90% CI the algorithms are statistically non-differentiable. In fact, it takes plotting the data at a 1% CI to differentiate a “winner” and a “loser” in the same plot as shown in Figure 4-15(b). Since the data obtained in this research can not be validated at the topology ranges of 8- to 128-nodes, a CI of 1% opens the research to risk of the actual data existing outside the range which is unacceptable.

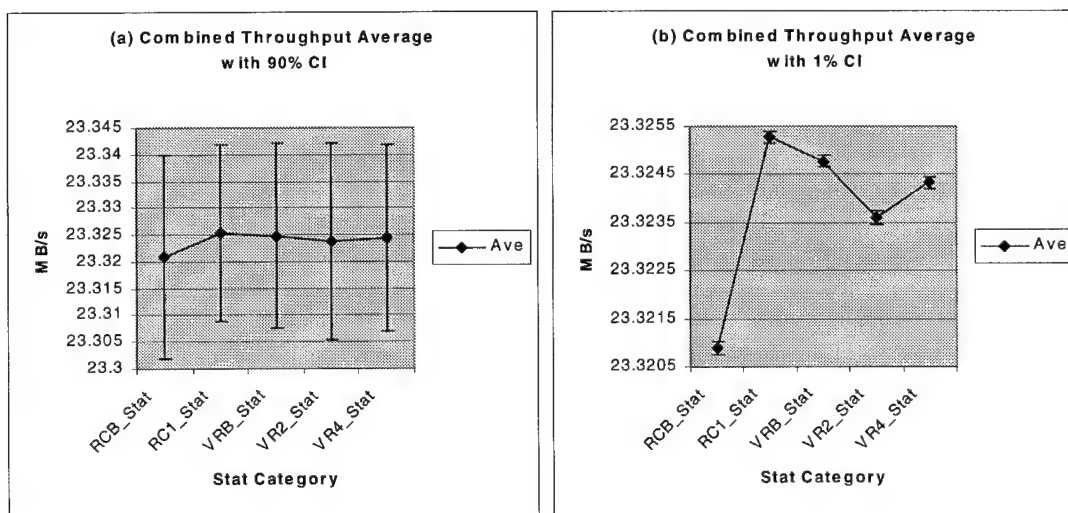


Figure 4-15 Combined Throughput for Stat Tests

Therefore, in an attempt to quantify which algorithm performs above the rest, Figure 4-16 shows the arithmetic mean and two weighted-means that the tests achieved. In the first view, Figure 4-16(a), the results achieved in the topology tests are scored based on a 1 to 5 scale where the highest performing algorithm receives a score of 5. Each remaining algorithm in that test is given a descending score of 4, 3, 2, or 1 depending on the order of performance. In cases where the performances were considered statistically equal, the points are combined and divided between the algorithms. For example, two are considered equivalent and are the highest performers so their scores are 4.5 each.

	RCB_Stat	RC1_Stat	VRB_Stat	VR2_Stat	VR4_Stat		RCB_Stat	RC1_Stat	VRB_Stat	VR2_Stat	VR4_Stat
2-Node	3	3	3	3	3	2-Node	0.696	0.696	0.696	0.696	0.696
4-Node	2.5	2.5	2.5	2.5	5	4-Node	0.580	0.580	0.580	0.580	1.161
8-Node	1	3.5	3.5	3.5	3.5	8-Node	0.107	0.375	0.375	0.375	0.375
16-Node	1	2.5	4.5	4.5	2.5	16-Node	0.107	0.268	0.482	0.482	0.268
32-Node	1	4.5	2.5	4.5	2.5	32-Node	0.107	0.482	0.268	0.482	0.268
64-Node	2	3.5	5	1	3.5	64-Node	0.214	0.375	0.536	0.107	0.375
128-Node	1	5	3	3	3	128-Node	0.107	0.536	0.321	0.321	0.321
Ave	1.643	3.500	3.429	3.143	3.286	Ave	0.274	0.473	0.466	0.435	0.495

(a)

	RCB_Stat	RC1_Stat	VRB_Stat	VR2_Stat	VR4_Stat
2-Node	0.107	0.107	0.107	0.107	0.107
4-Node	0.179	0.179	0.179	0.179	0.357
8-Node	0.107	0.375	0.375	0.375	0.375
16-Node	0.143	0.357	0.643	0.643	0.357
32-Node	0.179	0.804	0.446	0.804	0.446
64-Node	0.429	0.750	1.071	0.214	0.750
128-Node	0.250	1.250	0.750	0.750	0.750
Ave	0.199	0.546	0.510	0.439	0.449

(b)

(c)

Figure 4-16 Combined Results for the Stat Tests Using Arithmetic (a) and Weighted (b & c) Means

Thus, considering only the arithmetic means, the *Row Column Optimization 1* algorithm scores the highest with the *Vector Radix Basic* scoring the next highest. In Figure 4-16(b), the scores awarded in part (a) are given a weight where the 2- and 4-node tests are given the highest, equal values. Conversely, the remaining topologies were weighted equally as well. This breakout of weights places emphasis on the two test topologies that are

verifiable. As a result, the *Vector Radix Optimization 4* algorithm achieves the highest, combined rating with the *Row Column Optimization 1* algorithm scoring the next highest. The final plot weights the tests according to the size of the topology. In this scheme, the larger the number of nodes the higher the weight. The results using this method show the *Row Column Optimization 1* algorithm scoring the highest again and the *Vector Radix Basic* again scoring the next highest.

#### **4.4.9 Comparative Analysis Summary**

Considering each of the tests independently and the combined results of the tests, the conclusion reached by this research is that the *Row Column Optimization 1* algorithm outperforms the other algorithms when assuming the statistical approximation method. This algorithm was equal in performance to the others using the captured traffic patterns as inputs to the model. In addition, it consistently performed at throughput rates that placed it at the top of four of the statistical topology tests. When performance across the topology variations was considered, the *Row Column Optimization 1* algorithm scored best in the arithmetic mean classification and best when weighting the topologies with an increasing weight as the number of nodes was increased. In the third mean, the algorithms were weighted with the two validated tests carrying higher weights. With this scheme, the *Row Column Optimization 1* algorithm scored the second highest.

#### **4.5 Summary**

This chapter focused on analyzing the data generated by the test cases enumerated in Subsection 3.2.6. Section 4.2 analyzed the traffic and simulation run-time data while

Sections 4.3 and 4.4 analyzed the data resulting from the simulation test cases. In Subsection 4.4.9, the *Row Column Optimization 1* algorithm was identified as the best performing option in light of the data gathered.

However, the bandwidths achieved by the various algorithms all show an alarming problem. The top average bandwidth of 23.327 MB/s achieved the Row Column Optimized 1 algorithm is only 14.58% of the theoretical network maximum of 160 MB/s. An even bigger concern is seen when the MPI validation data is plotted next to the theoretical values as displayed in Figure 4-17. In this instance, the MPI data only achieves 11.13% of the maximum. In this chart, the theoretical maximum achievable by the Sun Sparc Ultra is provided along with a theoretical machine that transfers data at 160 MB/s and the model validation bandwidth data. The data was taken by considering the costs incurred to transfer from one system to another across one 6-meter cable at speeds of 1.28Gbps. First, a DMA from the system memory to the Myrinet adapter incurs the latency of the bus. Next, the worm encounters the latency associated with creating the route, flitizing the worm, and sending the appropriate control signals. Next, the worm's head must travel the length of the cable incurring an additional delay. Once the first flit arrives, the remaining flits follow in successive fashion at the maximum rate. The next penalty paid is at the destination end where the worm must be reassembled. Finally, the worm again incurs the DMA delay following re-assembly.

As a result of the costs to send a worm, the theoretical maximum bandwidth is greatly reduced from the 160MB/s rate to 55MB/s. Now that the maximum has been adjusted for fixed-costs, the model achieves 42.3% for the Row Column Optimized 1 algorithm.



Similarly, the MPI percentage increases to 32.4 percent. In both cases, one performance impediment is the SBus. Incurring this overhead twice, which on average is 34.38% of the 160 MB/s maximum, constrains the performance dramatically. In contrast, the PCI speed as tested in the simulations achieves 77.5% of the maximum. Thus, the necessity to switch to the PCI based systems is self-evident.

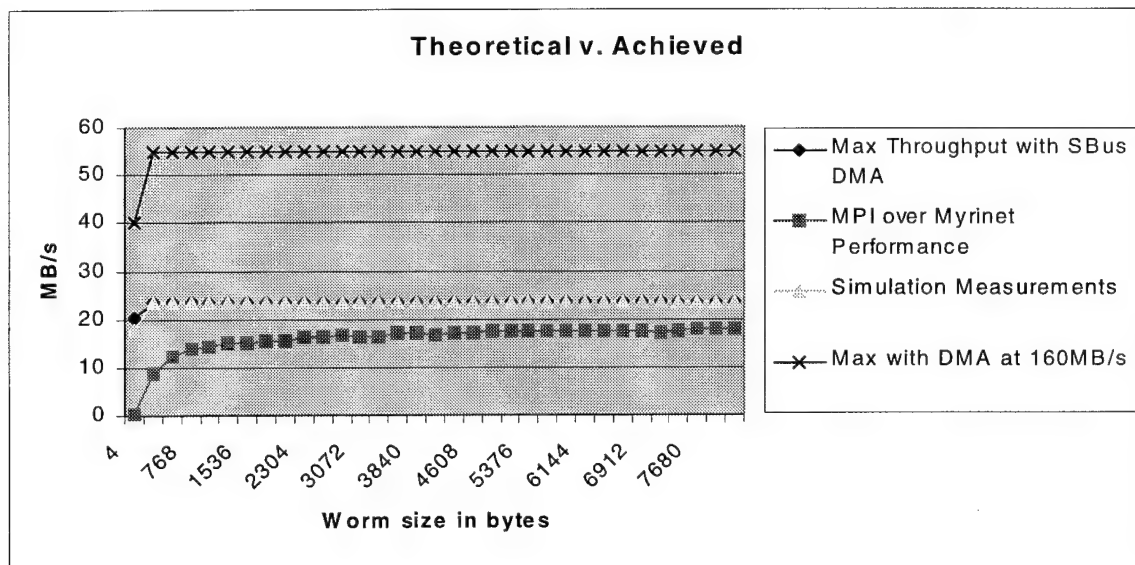


Figure 4-17 Theoretical versus Achieved Bandwidth

Beyond the hardware problems is the even greater concern of the MPI penalty. As displayed in Figure 4-17, the MPI further reduces the performance from the adjusted maximum by ten percent. The main factor in the reduction in performance is seen during a traffic trace of an MPI program. Depending on the type of communication command executed, the MPI environment generates at least two worms to complete one data segment. As Figure 4-18 shows, the resulting overhead averaged nearly 75% of the worms generated during a trace.

<b>Trace</b>	<b>% Overhead Messages</b>	<b>% Overhead Bytes Communicated</b>
RC Basic 2-Node	71.34%	2.36%
RC Basic 4-Node	80.08%	4.21%
RC Opt 1 2-Node	71.43%	2.36%
RC Opt 1 4-Node	80.70%	4.33%
VR Basic 2-Node	71.60%	2.38%
VR Basic 4-Node	78.17%	3.79%
VR Opt 2 2-Node	67.01%	1.98%
VR Opt 2 4-Node	78.38%	3.86%
VR Opt 4 2-Node	71.34%	2.36%
VR Opt 4 4-Node	77.84%	3.73%
<b>Average</b>	<b>74.79%</b>	<b>3.13%</b>

*Figure 4-18 Percentage Overhead Observed During Traffic Traces*

Furthermore, the amount of overhead increased in each case when the number of nodes increased. The biggest problem was the fact that MPI generated a third worm as part of the communication acknowledgement scheme. The two overhead worms are 86 bytes or less which was shown to be a big performance bottleneck due to the small size. Some instances of parallel programs running on MPI are able to take advantage of communicating large data chunks, e.g. data size greater than the 8KB Myrinet TCP/IP limit. The resulting process to break-up the chunk avoids some of the overhead that would result from having the program send multiple messages to accomplish the same chunk size. However, as shown in Figure 4-18, the overhead messages accounted for only 3.13% of the total bytes communicated on average during the traffic traces. So, MPI generates a lot of small messages to accomplish the parallel communications during a program such as this class of FFT algorithms. It is this tendency toward small messages that poses the greatest problem to achieving the theoretical maximum bandwidth. As noted in Chapter 2, the common case is

for worm sizes smaller than the 8KB Myrinet MTU which, the Fast Messages research identified as one of the biggest challenges to a network such as Myrinet [PaK97] [LaC97]. Therefore, until the messaging layers are streamlined and operationally sound, achieving the adjusted theoretical maximum will continue as nothing more than an academic exercise.

## ***V. Conclusion***

### ***5.1 Review of Thesis***

This chapter culminates this thesis effort by briefly reviewing the information presented. In addition, Section 5.2 summarizes the conclusions drawn as a result of the data collected while Section 5.3 outlines some recommendations derived from the research.

In Chapter 1, a background on the problem and a plan of attack to research a solution to the problem were provided. The background information necessary to conduct the research was discussed in Chapter 2. Issues such as message passing, hardware support, and traffic analysis were explored. Key areas focused on the Myrinet and its technical characteristics necessary to create a model of and simulate the performance of a Myrinet.

Chapter 3 provided the methodology used in this research effort. The details of the model were presented as well as the process used to analyze actual system data. The problem was further scoped and defined to better organize the thesis. The chapter also provided the verification and validation process while outlining the actual test cases executed to gather the data.

The data captured from the test cases was analyzed in Chapter 4. First, an analysis of the traffic patterns and an analysis of the computer resources consumed during the simulations were provided. Then, actual performance comparisons were made on the algorithms' performance under the captured data and under the statistical representation method. From the analysis provided in Chapter 4, conclusions for this thesis effort are summarized in Section 5.2.

## ***5.2 Summary of Conclusions***

As Chapter 4 stated, the Myrinet model developed in this thesis provided a solid simulation environment to test parallel algorithms, which happened to be Fast Fourier Transforms (FFTs) in this case. Four validation tests showed that the model simulates the performance achieved by an actual Myrinet with a maximum .045 percent difference. Furthermore, the model simulated five different FFTs using traffic analysis that showed the communications patterns were bursty.

As stated, the model simulated five different algorithms across seven different topologies. The tests showed that when considering the topologies independently using a 90% confidence interval (CI), only one algorithm consistently performed below the others, *Row Column Basic*. The others each had at least one topology where the algorithms performed at the highest level. In contrast, when aggregating an algorithm's performance across all of the topologies, the 90% CI did not differentiate one algorithm from the others. In fact, the only CI that did differentiate both a statistically superior algorithm and a statistically inferior algorithm was the 5% CI.

Finally, Chapter 4 showed that the MPI overhead constrained the performance of the algorithms. In fact, traffic analysis data displayed the fact that 75% of the worms generated during the traffic traces were smaller than 100 bytes. These small messages are the greatest impediment to performance within a high-speed network such as Myrinet.

## ***5.3 Recommendations for Future Work***

This research provided a framework for advanced modeling and simulation of Myrinet networks to study parallel processing. An investigation of five different algorithms

was provided as a basis for the proof of the model. As intense as the simulations were, consuming two man months and stressing a network of 25 advance workstations, many areas are left to study. Thus, the following provides a basis to build future work from:

- *Validate the results achieved by higher order topologies.* Limited resources notwithstanding, this would prove the model's accuracy even more.
- *Investigate other traffic sources as inputs to the model.* Due to limited time and manning, other algorithms were not considered as inputs to the model. Future work should include other parallel algorithms to open the model to further validation. Also, other parallel platforms could be monitored to compare the performance achieved on those with the performance achieved by the Myrinet model.
- *Research improvements to the Adapter.* The model's current use of the route map requires the user to develop and to place the route in the model as an input parameter. The algorithm employed in a Myrinet is termed "complex" by Myricom but could be coded in to Designer through C/C++.
- *Investigate Designer enhancements.* A better queue algorithm to handle block deletes could be developed in C/C++, which could dramatically improve simulation run-time.

In conclusion, this thesis has shown a model for simulating higher-order topologies in a Myrinet that achieves very similar performance to actual systems. While further work is needed to open the model to a wider range of applications, the initial results have merit. The greatest roadblock to achieving high bandwidth was shown to be the MPI overhead incurred for communicating the parallel traffic. Finally, this research showed that the class of FFT algorithms under investigation met the definition for bursty traffic. As a result, this work serves as a solid foundation to build future advanced networks based on high-speed, cut-through routing switches.

## **Appendix A: Statistical Analysis**

### ***A.1 Overview***

This appendix provides the full range of statistics computed for the traffic analysis and for the simulations. Section A.2 provides the variance-time plot analysis for the Myrinet traffic traces. Following the traffic analysis, Section A.3 provides the scaled traffic statistics for each of the algorithms investigated. Section A.4 lists the data for determining the number of independent.

### ***A.2 Variance-Time Plot Analysis***

Determining self-similarity requires monitoring traffic as it is communicated over a network. The monitored data is recorded for a temporal distribution allowing the number of worms received during a time window to be counted. Each of the five FFT algorithms was tracked to record data across the entire execution time of the program. Statistical analysis was performed for each algorithm to support a 99% confidence interval with an error of +/- .001 percent. The following subsections display data across time windows that range from .0001 to .1 second intervals. In some cases, the amount of data produced was too large to plot using Microsoft Excel requiring some plots to be dropped. In all cases, the variance-time plot analysis seeks to identify visually the degree of self-similarity. If a series of plots did not appear to be similar, then further statistical analysis would be required to support a "bursty" claim.

## A.2.1 RCB\_256\_4 Trace

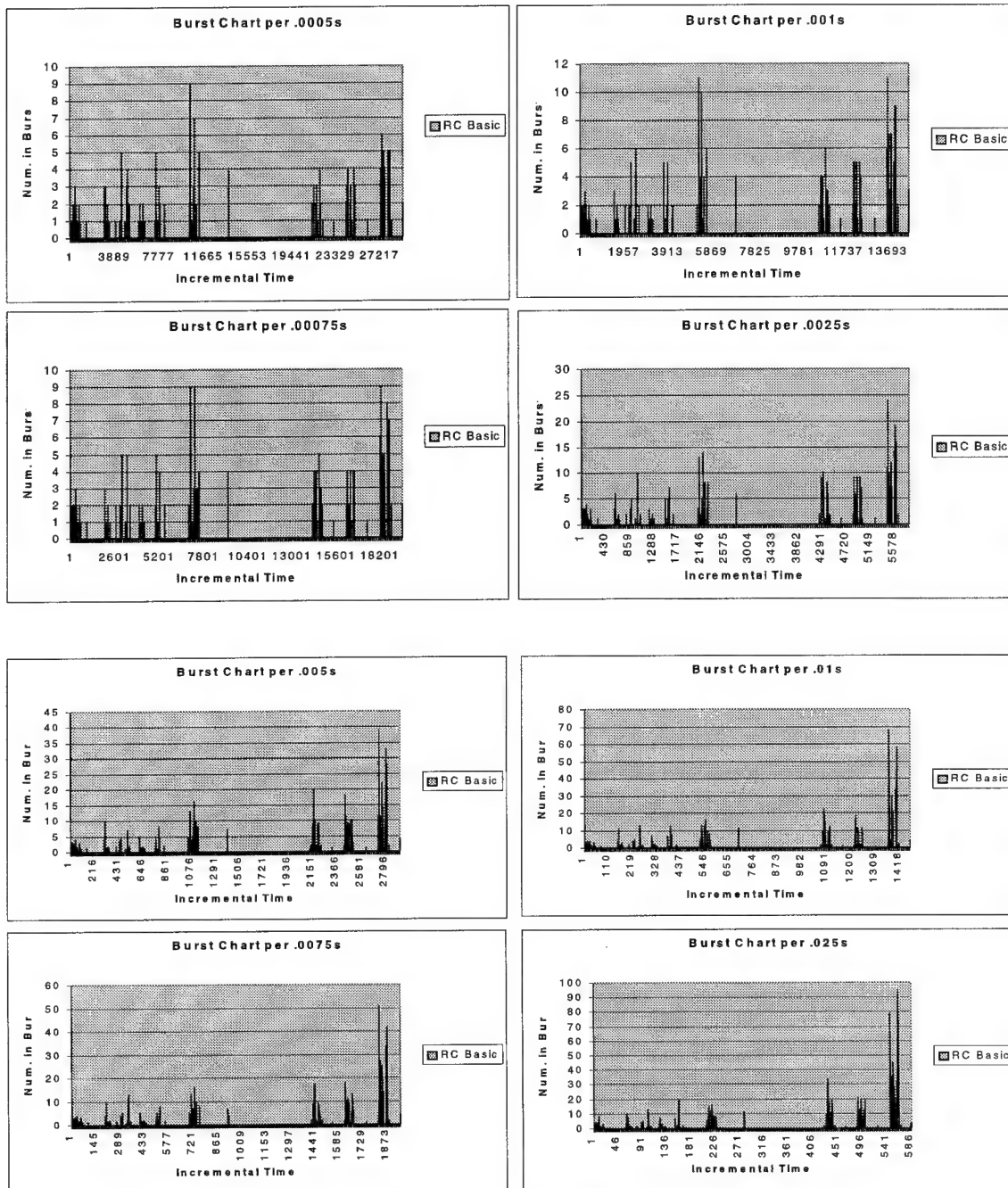


Figure A-1 RCB Basic 4-Node Variance-Time Plot for .0005 to .025 seconds



## A.2.2 RC1\_256\_2 Trace

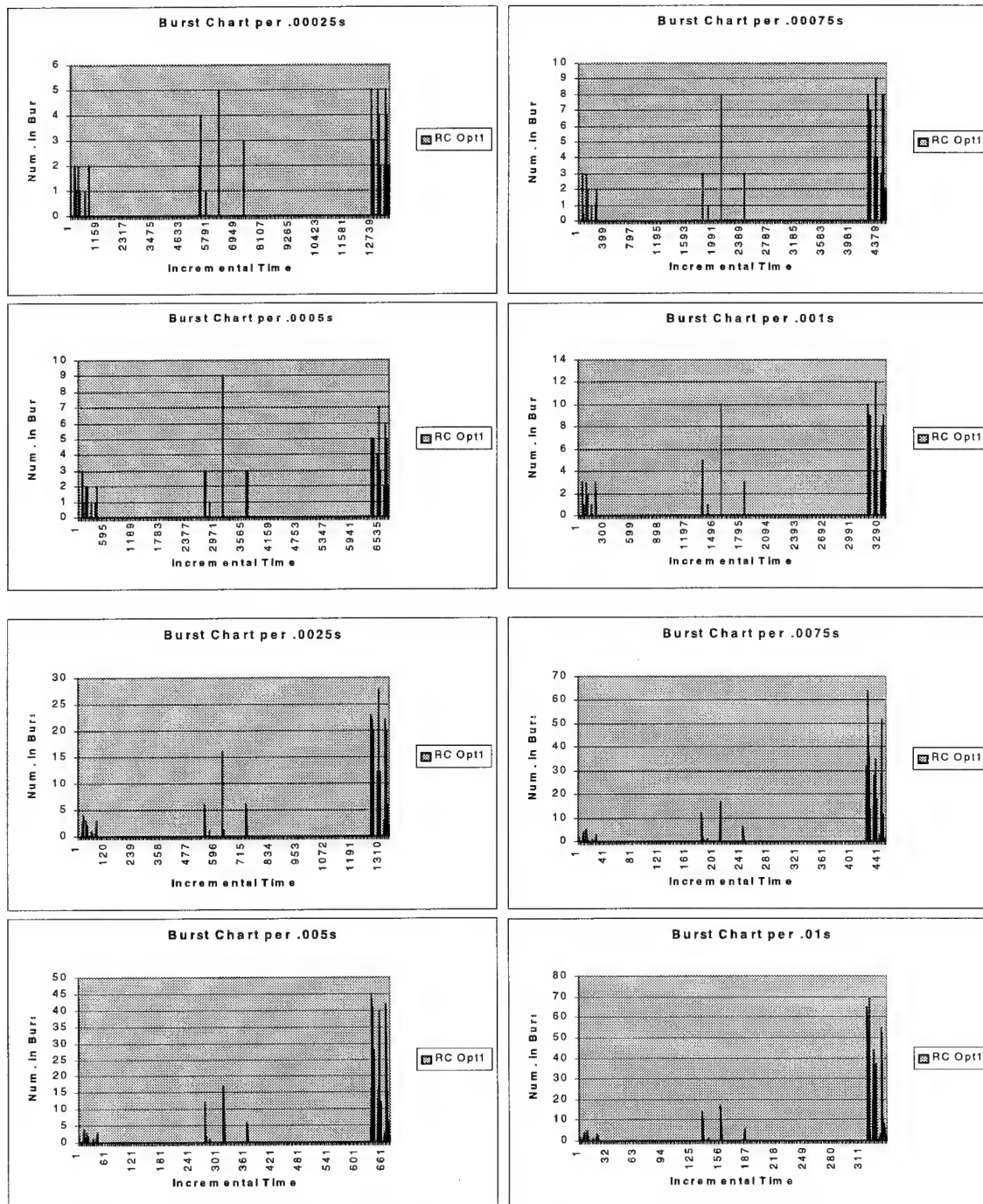


Figure A-2 RC Opt1 2-Node Variance-Time Plot for .00025 to .01 seconds

### A.2.3 RC1\_256\_4 Trace

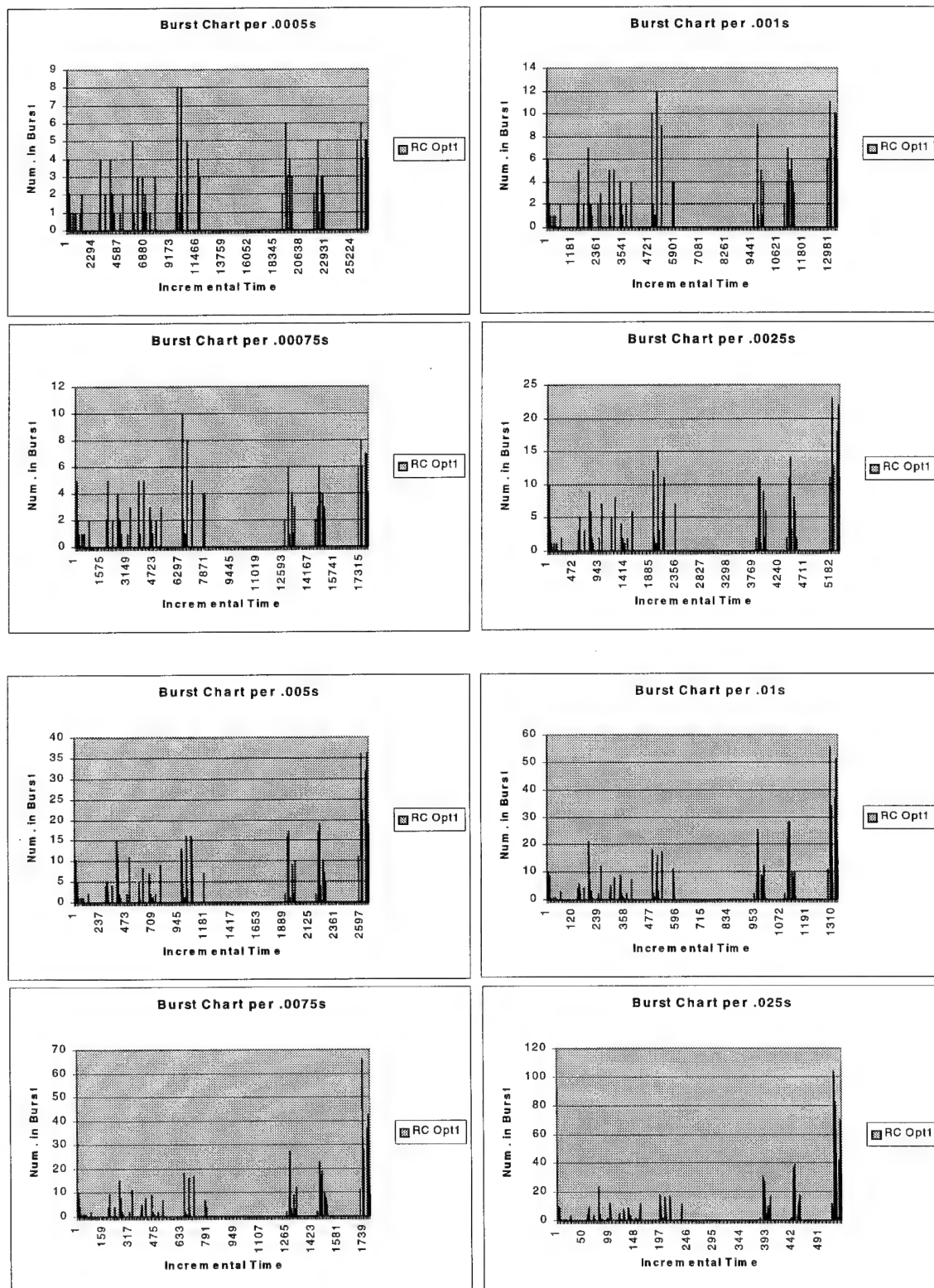


Figure A-3 RC Opt1 4-Node Variance-Time Plot .0005 to .025 seconds

## A.2.4 VRB\_256\_2 Trace

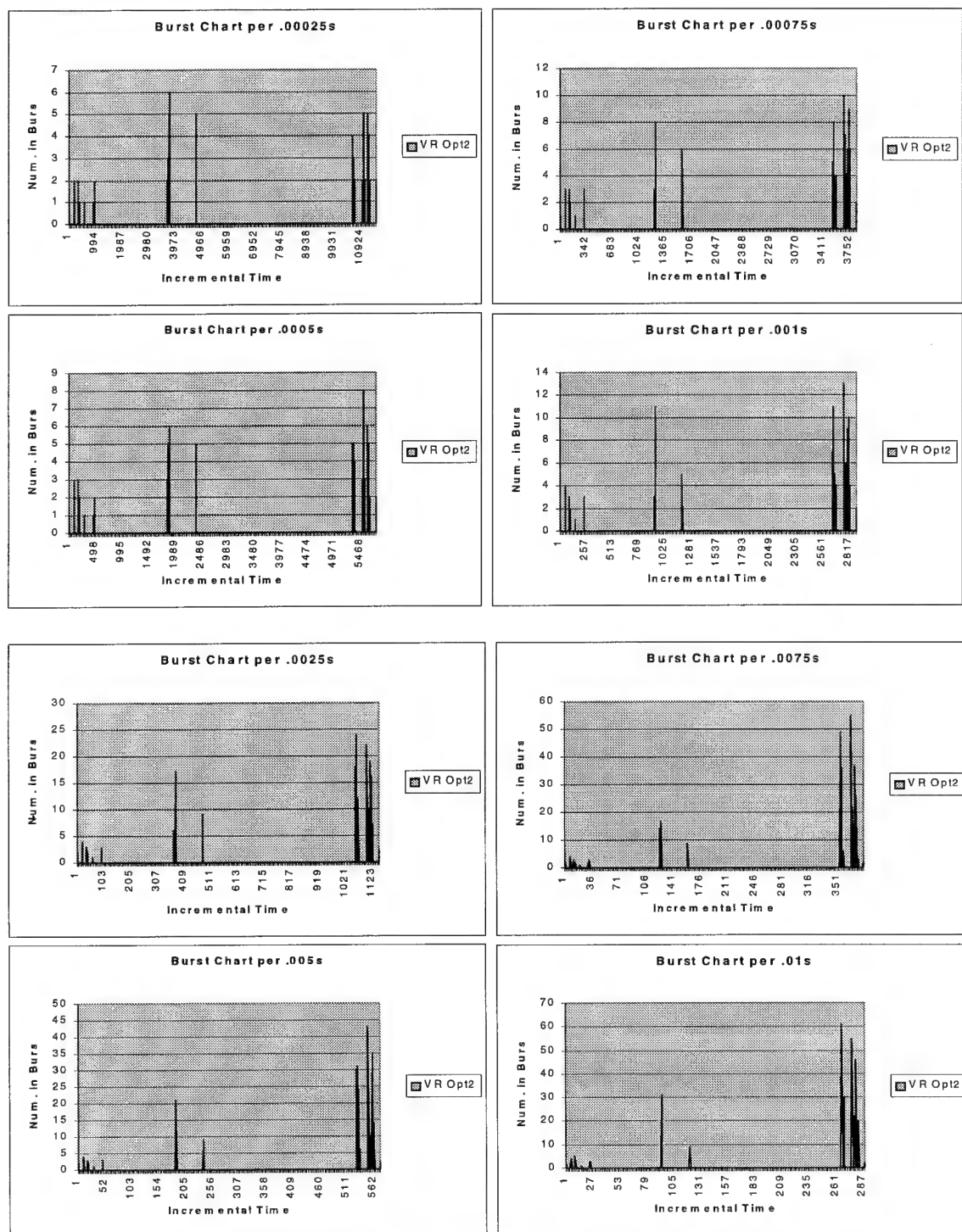


Figure A-4 VR Basic 2-Node Variance-Time Plot .00025 to .01 seconds

## A.2.5 VRB\_256\_4 Trace

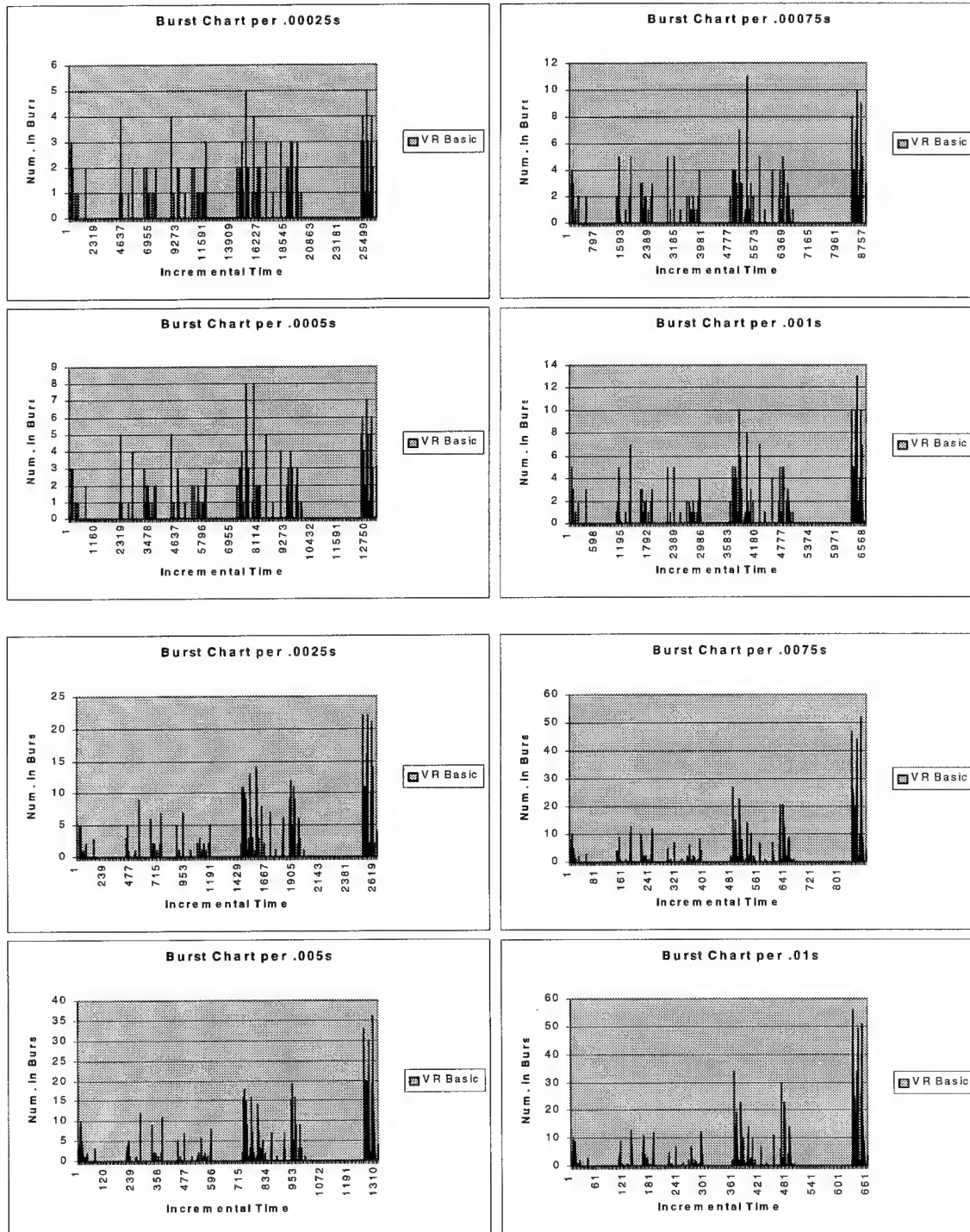


Figure A-5 VR Basic 4-Node Variance-Time Plot .00025 to .01 seconds

## A.2.6 VR2\_256\_2 Trace

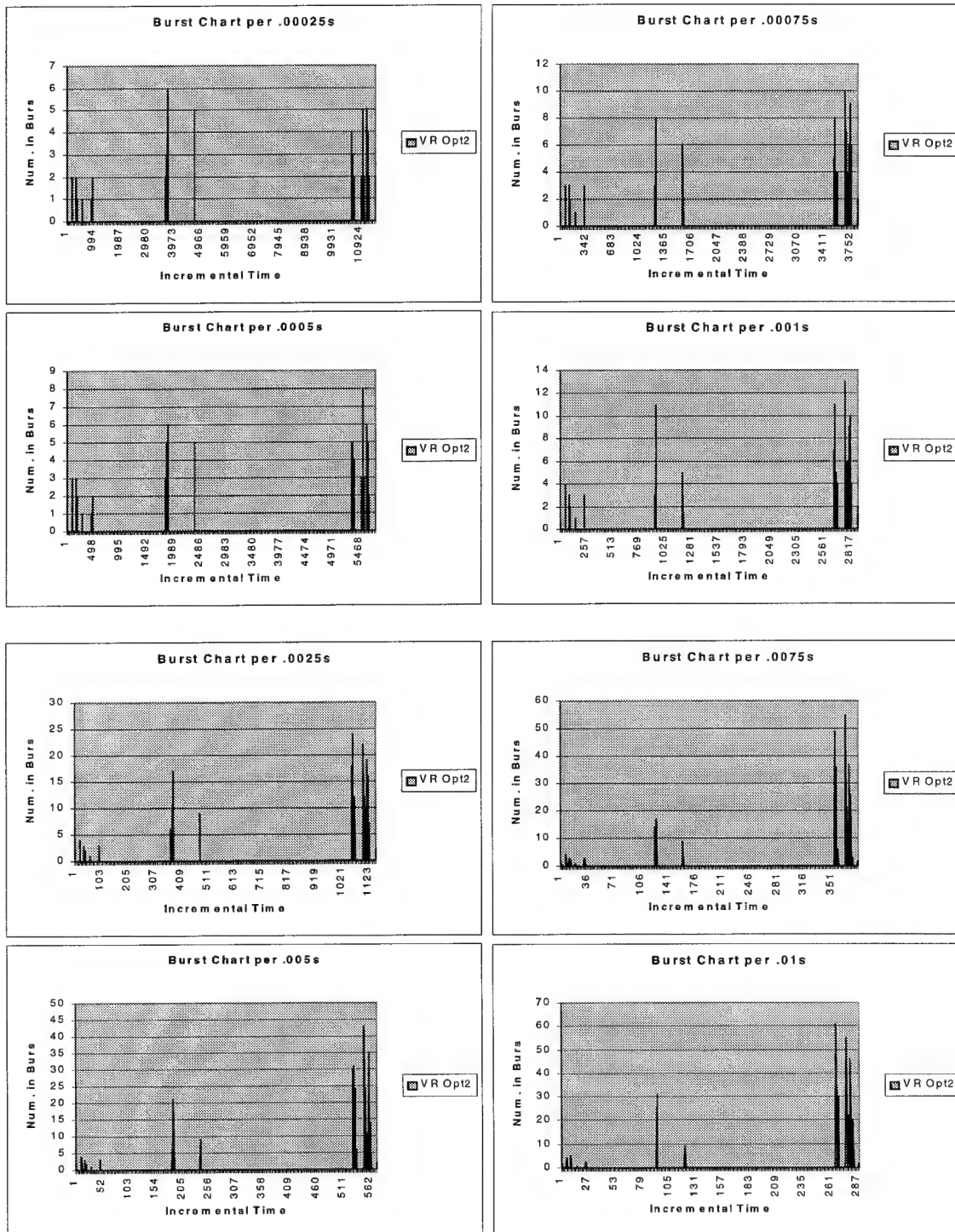


Figure A-6 VR Opt2 2-Node Variance-Time Plot .00025 to .01 seconds



## A.2.7 VR2\_256\_4 Trace

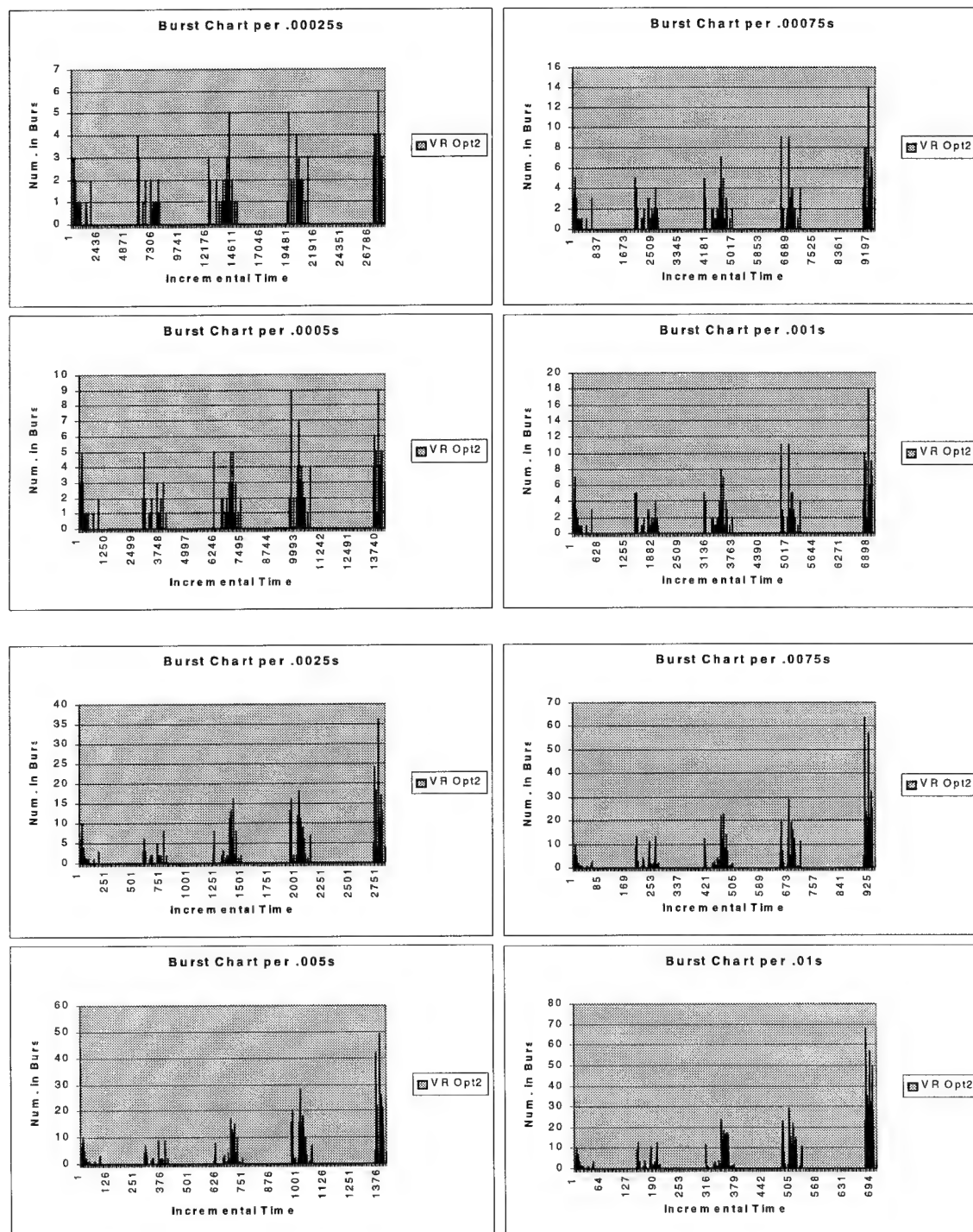


Figure A-7 VR Opt2 4-Node Variance-Time Plot .00025 to .01 seconds

## A.2.8 VR4\_256\_2 Trace

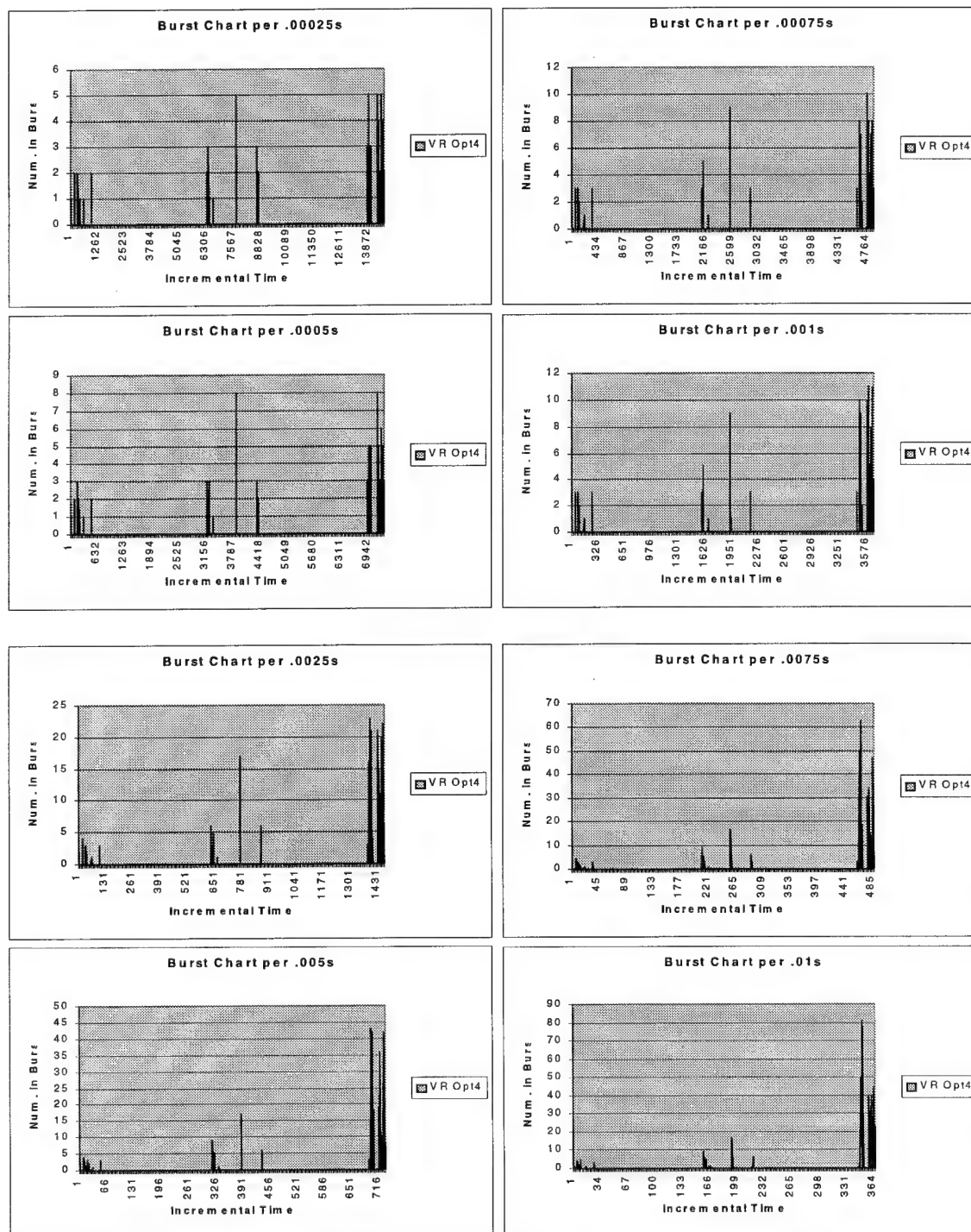


Figure A-8 VR Opt4 2-Node Variance-Time Plot.00025 to .01 seconds

### A.2.9 VR4\_256\_4 Trace

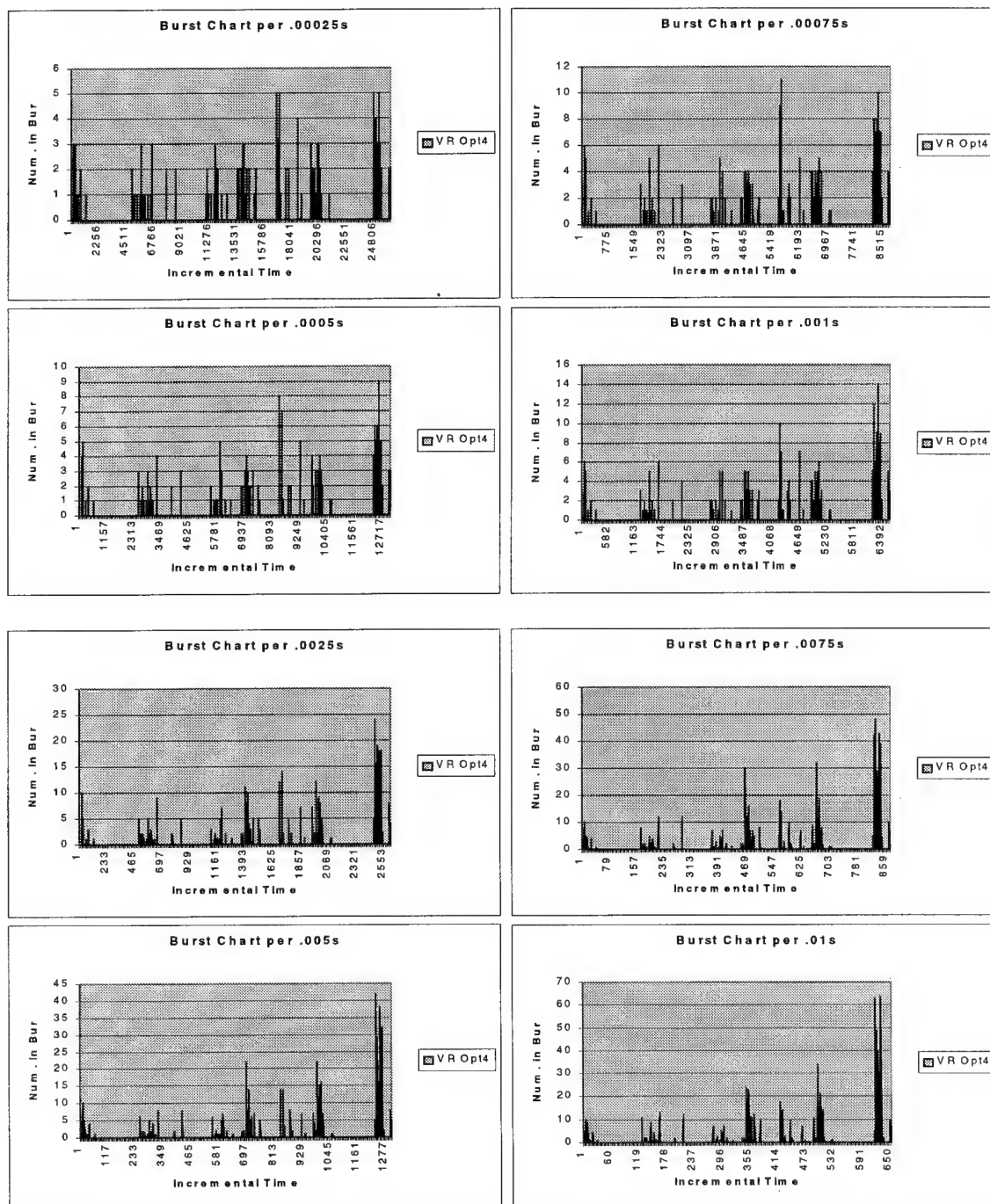


Figure A-9 VR Opt4 4-Node Variance-Time Plot .00025 to .01 seconds



### A.3 Scaled Traffic Statistics

The scaled traffic statistics represent the values used in the “bursty” traffic generator for each of the algorithms investigated. The data is a simple scaling where the difference between the two node and the four node versions of each algorithms is scaled as the degree of the topology increases.

#### A.3.1 RC Basic

Con Data								
	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.00	1.00	1.00	1.00	1.00
Ave. Number in Burst	24.50	11.50	-53.06%	9.974490	9.312916	9.004070	8.854768	8.707941
Ave. Delay Btwn Pkts	0.05	0.04	-21.46%	0.038983	0.037937	0.037428	0.037177	0.036927
Size Variance	34602898.94	8205659.97	-76.29%	6640713.289458	6007469.757083	5721040.332728	5584653.937500	5451518.917499
Ave. Size	8125.84	8104.58	-0.26%	8099.274832	8096.625938	8095.301924	8094.640026	8093.978181

Con Overhead								
	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.00	1.00	1.00	1.00	1.00
Ave. Number in Burst	32.95	29.60	-10.17%	28.848054	28.481176	28.300069	28.210092	28.120401
Ave. Delay Btwn Pkts	0.08	0.33	332.36%	0.599714	0.848862	1.025189	1.131667	1.249203
Size Variance	51825.11	266524.30	414.28%	542561.103925	823523.921240	1036752.992918	1170972.456699	1322568.156267
Ave. Size	69.56	97.26	39.82%	106.940311	112.263762	115.057987	116.489873	117.939579

Worker Data								
	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.000000	1.000000	1.000000	1.00	1.00
Ave. Number in Burst	24.50	9.83	-59.86%	8.361678	7.735974	7.446533	7.307227	7.170527
Ave. Delay Btwn Pkts	0.02	0.06	191.29%	0.092239	0.114295	0.127960	0.135609	0.143716
Size Variance	41323779.29	18461061.45	-55.33%	15907628.259862	14807500.129667	14295476.993207	14048317.955451	13805432.126723
Ave. Size	8141.24	8204.57	0.78%	8220.523381	8228.516588	8232.517227	8234.518469	8236.520198

Worker Overhead								
	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.000000	1.000000	1.000000	1.00	1.00
Ave. Number in Burst	45.47	22.58	-50.35%	19.734510	18.492487	17.910560	17.628752	17.351379
Ave. Delay Btwn Pkts	0.06	0.18	179.41%	0.258579	0.316569	0.352067	0.371806	0.392652
Size Variance	36453.30	215093.21	490.05%	478610.005383	771789.355929	1008174.616982	1162567.534033	1340604.344150
Ave. Size	60.25	69.79	15.83%	72.550718	73.986042	74.717903	75.087453	75.458831

Figure A-10 VR Basic Controller and Worker Statistics

### A.3.2 VR Optimization 1

#### Con Data

	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.00	1.00	1.00	1.00	1.00
Ave. Number in Burst	24.50	10.50	-57.14%	9.000000	8.357143	8.058673	7.914769	7.773433
Ave. Delay Btwn Pkts	0.04	0.07	72.96%	0.080740	0.088103	0.092120	0.094220	0.096368
Size Variance	34602898.94	7399021.78	-78.62%	5944793.334464	5360588.569216	5097191.575370	4971964.194022	4849813.388629
Ave. Size	8125.84	8094.12	-0.39%	8096.222558	8082.277373	8080.305743	8079.320168	8078.334714

#### Con Overhead

	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.00	1.00	1.00	1.00	1.00
Ave. Number in Burst	32.06	18.58	-42.05%	16.622533	15.748725	15.334788	15.133259	14.934379
Ave. Delay Btwn Pkts	0.08	0.18	133.84%	0.245002	0.285989	0.309912	0.322873	0.336377
Size Variance	52414.74	151082.17	188.24%	222182.825344	274463.460894	306754.715065	324799.912119	343906.638535
Ave. Size	69.52	90.81	30.62%	97.764733	101.506911	103.449621	104.439566	105.438984

#### Worker Data

	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.000000	1.000000	1.000000	1.00	1.00
Ave. Number in Burst	24.50	9.60	-60.84%	8.135904	7.517212	7.231389	7.093912	6.959049
Ave. Delay Btwn Pkts	0.04	0.11	179.93%	0.154157	0.188828	0.210063	0.221874	0.234349
Size Variance	41323779.29	19000092.62	-54.02%	16434063.316463	15324324.303922	14906923.336389	14556957.455072	14311211.420134
Ave. Size	8141.24	8196.92	0.68%	8210.933321	8217.952907	8221.465700	8223.222847	8224.980370

#### Worker Overhead

	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.000000	1.000000	1.000000	1.00	1.00
Ave. Number in Burst	45.47	21.42	-52.90%	18.584760	17.355876	16.782063	16.504642	16.231807
Ave. Delay Btwn Pkts	0.06	0.15	166.56%	0.214449	0.259098	0.286070	0.300960	0.316625
Size Variance	36453.30	105167.86	188.50%	154728.268440	191186.158245	213710.303186	226299.190082	239629.641941
Ave. Size	60.25	65.57	8.82%	67.013167	67.751988	68.125471	68.313242	68.501531

Figure A-11 RC Opt1 Controller and Worker Statistics

### A.3.3 VR Basic

#### Con Data

	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.00	1.00	1.00	1.00	1.00
Ave. Number in Burst	24.50	11.50	-53.06%	9.974490	9.312916	9.004070	8.854768	8.707941
Ave. Delay Btwn Pkts	0.05	0.04	-21.46%	0.038983	0.037937	0.037428	0.037177	0.036927
Size Variance	34602898.94	8205659.97	-76.29%	6640713.289458	6007469.757083	5721040.332728	5584653.937500	5451518.917499
Ave. Size	8125.84	8104.58	-0.26%	8099.274832	8096.625938	8095.301924	8094.640026	8093.978181

#### Con Overhead

	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.00	1.00	1.00	1.00	1.00
Ave. Number in Burst	32.95	29.60	-10.17%	28.848054	28.481176	28.300069	28.210092	28.120401
Ave. Delay Btwn Pkts	0.08	0.33	332.36%	0.599714	0.848862	1.025189	1.131667	1.249203
Size Variance	51825.11	266524.30	414.28%	542561.103925	823523.921240	1038752.992918	1170972.456699	1322568.156267
Ave. Size	69.56	97.26	39.82%	106.940311	112.263762	115.057987	116.489673	117.939579

#### Worker Data

	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.000000	1.000000	1.000000	1.00	1.00
Ave. Number in Burst	24.50	9.83	-59.86%	8.361678	7.735974	7.446533	7.307227	7.170527
Ave. Delay Btwn Pkts	0.02	0.06	191.29%	0.092239	0.114295	0.127960	0.135609	0.143716
Size Variance	41323779.29	18461061.45	-55.33%	15907628.259862	14807500.129667	14295476.993207	14048317.955451	13805432.128723
Ave. Size	8141.24	8204.57	0.78%	8220.523381	8228.516688	8232.517227	8234.518469	8236.520198

#### Worker Overhead

	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.000000	1.000000	1.000000	1.00	1.00
Ave. Number in Burst	45.47	22.58	-50.35%	19.734510	18.492487	17.910560	17.628752	17.351379
Ave. Delay Btwn Pkts	0.06	0.18	179.41%	0.258579	0.316569	0.352067	0.371806	0.392652
Size Variance	36453.30	215093.21	490.05%	478610.005383	771789.355929	1008174.616982	1162567.534033	1340604.344150
Ave. Size	60.25	69.79	15.83%	72.550718	73.986042	74.717903	75.087453	75.458931

Figure A-12 VR Basic Controller and Worker Statistics

### A.3.4 VR Optimization 2

#### Con Data

	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.00	1.00	1.00	1.00	1.00
Ave. Number in Burst	24.50	11.50	-53.06%	9.974490	9.312916	9.004070	8.854768	8.707941
Ave. Delay Btwn Pkts	0.05	0.03	-30.26%	0.031865	0.030660	0.030080	0.029796	0.029514
Size Variance	35049636.01	8053642.72	-77.02%	6502869.949985	5876788.435677	5593886.529951	5459244.864944	5327843.948183
Ave. Size	8137.92	8089.83	-0.59%	8077.880847	8071.914336	8068.933285	8067.443309	8065.953609

#### Con Overhead

	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.00	1.00	1.00	1.00	1.00
Ave. Number in Burst	23.54	25.94	10.18%	26.599097	26.937713	27.109177	27.195454	27.282006
Ave. Delay Btwn Pkts	0.12	0.13	12.92%	0.136506	0.138710	0.139830	0.140395	0.140962
Size Variance	126104.46	109873.17	-12.87%	106337.639265	104626.758630	103785.081608	103367.628560	102951.854627
Ave. Size	76.24	83.65	9.71%	85.680420	86.720783	87.247280	87.512127	87.777778

#### Worker Data

	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.000000	1.000000	1.000000	1.00	1.00
Ave. Number in Burst	24.50	9.72	-60.32%	8.256173	7.633684	7.345906	7.207441	7.071587
Ave. Delay Btwn Pkts	0.02	0.04	100.92%	0.053514	0.060265	0.064066	0.068086	0.068170
Size Variance	41323779.29	18335084.69	-55.63%	15785102.152729	14687432.424682	14176762.623520	13930305.504918	13688132.940762
Ave. Size	8141.24	8203.48	0.76%	8219.163022	8227.018091	8230.949378	8232.915961	8234.883014

#### Worker Overhead

	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.000000	1.000000	1.000000	1.00	1.00
Ave. Number in Burst	35.43	21.32	-39.83%	19.193500	18.237872	17.783848	17.562487	17.343882
Ave. Delay Btwn Pkts	0.09	0.08	-6.51%	0.081592	0.080929	0.080599	0.080436	0.080272
Size Variance	88855.72	82387.15	-7.28%	80887.729229	80151.664720	79796.981496	79605.469523	79424.370482
Ave. Size	65.51	64.51	-1.53%	64.258489	64.135412	64.073992	64.043311	64.012645

Figure A-13 VR Opt2 Controller and Worker Statistics

### A.3.5 VR Optimization 4

Con Data								
	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.00	1.00	1.00	1.00	1.00
Ave. Number in Burst	24.50	11.50	-53.06%	9.974490	9.312916	9.004070	8.854768	8.707941
Ave. Delay Btwn Pkts	0.05	0.03	-42.58%	0.026158	0.024765	0.024106	0.023785	0.023469
Size Variance	34602898.94	8235032.65	-76.20%	6666231.720290	6031262.136609	5744018.295662	5607236.487779	5473711.853187
Ave. Size	8125.84	8109.25	-0.20%	8105.111781	8103.043727	8102.009964	8101.493148	8100.976365
Con Overhead								
	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.00	1.00	1.00	1.00	1.00
Ave. Number in Burst	31.61	28.04	-11.30%	27.246336	26.861550	26.671874	26.577705	26.483869
Ave. Delay Btwn Pkts	0.08	0.27	226.79%	0.420603	0.539837	0.616355	0.660036	0.706813
Size Variance	52718.84	296819.23	463.02%	640404.577776	1011057.168585	1303646.394207	1492277.116989	1708201.705449
Ave. Size	70.59	98.19	39.10%	107.764446	113.052209	115.814817	117.229875	118.652223
Worker Data								
	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.000000	1.000000	1.000000	1.00	1.00
Ave. Number in Burst	24.50	9.72	-60.31%	8.257227	7.634705	7.346910	7.208437	7.072574
Ave. Delay Btwn Pkts	0.02	0.06	138.48%	0.074163	0.087001	0.094531	0.098622	0.102890
Size Variance	41323779.29	19168324.64	-53.61%	16599083.909064	15486648.596000	14967707.464261	14716931.481230	14470357.116503
Ave. Size	8141.24	8194.97	0.66%	8208.493424	8215.265578	8218.654448	8220.349583	8222.045066
Worker Overhead								
	X2	X4	%+/-	X8	X16	X32	X64	X128
Burst Inter-arrival Time	1.00	1.00	0.00%	1.000000	1.000000	1.000000	1.00	1.00
Ave. Number in Burst	45.47	21.93	-51.76%	19.094983	17.859461	17.281671	17.002123	16.727096
Ave. Delay Btwn Pkts	0.07	0.18	172.09%	0.254317	0.309023	0.342261	0.360667	0.380063
Size Variance	36453.30	191895.68	426.42%	396463.690419	607786.298006	769767.065039	872342.147796	988585.842890
Ave. Size	60.25	68.94	14.42%	71.430685	72.718646	73.374238	73.704989	74.037231

Figure A-14 VR Opt4 Controller and Worker Statistics

#### A.4 Statistical Analysis for Number of Independent Replications

As described in Chapter 3, the statistical analysis of the simulations included determining the number of independent replications required for each test. The process to accomplish this required running each simulation until a steady-state was first determined and, then by determining the appropriate confidence interval to follow. In the following subsections, each algorithm's analysis is provided with the 90, 95, and 99% confidence intervals and a one percent error. The data is provided only on the "stat" simulations as the "form" simulations have no inherent variability. This lack of variability results in small numbers when calculating the number of replications.

##### A.4.1 RC Basic

n at 1% with 100 samples w/ z = 1.645				n at 1% with 100 samples w/ z = 1.96				n at 1% with 100 samples w/ z = 2.576			
	Worm Size	Latency	Throughput	Worm Size	Latency	Throughput		Worm Size	Latency	Throughput	
2 Nodes	Mean	7034.62	0.0003012	23.350179	7034.62	0.0003012	23.350179	7034.62	0.0003012	23.350179	
	StanDev	1995.9754	8.541E-05	0.014555	1995.9754	8.541E-05	0.014555	1995.9754	8.541E-05	0.014555	
	n	2178.5167	2175.5416	0.0105141	3092.7245	3088.5009	0.0149264	5342.2081	5334.9126	0.025783	
4 Nodes	Mean	6841.98	0.000293	23.349244	6841.98	0.000293	23.349244	6841.98	0.000293	23.349244	
	StanDev	2064.5513	8.834E-05	0.0153582	2064.5513	8.834E-05	0.0153582	2064.5513	8.834E-05	0.0153582	
	n	2463.8801	2460.4204	0.0117076	3497.8398	3492.9282	0.0166207	6041.9829	6033.4989	0.0287097	
8 Nodes	Mean	7123.38	0.000305	23.351305	7123.38	0.000305	23.351305	7123.38	0.000305	23.351305	
	StanDev	1784.2462	7.635E-05	0.0118698	1784.2462	7.635E-05	0.0118698	1784.2462	7.635E-05	0.0118698	
	n	1697.732	1695.4422	0.0069919	2410.18	2406.9292	0.0099261	4163.217	4157.6018	0.0171458	
16 Nodes	Mean	6962.41	0.0002984	23.327867	6962.41	0.0002984	23.327867	6962.41	0.0002984	23.327867	
	StanDev	2067.7111	8.846E-05	0.0331857	2067.7111	8.846E-05	0.0331857	2067.7111	8.846E-05	0.0331857	
	n	2386.6698	2378.7238	0.0547624	3388.2284	3376.9479	0.0777433	5852.6459	5833.1607	0.1342897	
32 Nodes	Mean	6537.35	0.0002804	23.303477	6537.35	0.0002804	23.303477	6537.35	0.0002804	23.303477	
	StanDev	2160.0518	9.241E-05	0.0558219	2160.0518	9.241E-05	0.0558219	2160.0518	9.241E-05	0.0558219	
	n	2954.3141	2940.0488	0.1552746	4194.0828	4173.8312	0.2204351	7244.6361	7209.6545	0.3807678	
64 Nodes	Mean	6668.52	0.0002861	23.293569	6668.52	0.0002861	23.293569	6668.52	0.0002861	23.293569	
	StanDev	2224.37	9.516E-05	0.063762	2224.37	9.516E-05	0.063762	2224.37	9.516E-05	0.063762	
	n	3010.8349	2994.3739	0.2027606	4274.3225	4250.9536	0.2878484	7383.2378	7342.8717	0.497214	
128 Nodes	Mean	6789.2	0.0002914	23.274475	6789.2	0.0002914	23.274475	6789.2	0.0002914	23.274475	
	StanDev	2251.329	9.627E-05	0.0951805	2251.329	9.627E-05	0.0951805	2251.329	9.627E-05	0.0951805	
	n	2975.5862	2954.3124	0.4525511	4224.2817	4194.0805	0.6424627	7296.8	7244.6321	1.109756	

Figure A-15 RC Basic Number of Independent Replications Statistics

## A.4.2 RC Opt1

n at 1% with 100 samples w/ z = 1.645				n at 1% with 100 samples w/ z = 1.96				n at 1% with 100 samples w/ z = 2.576			
	Worm Size	Latency	Throughput	Worm Size	Latency	Throughput		Worm Size	Latency	Throughput	
2 Nodes	Mean	6803.01	0.0002913	23.348776	6803.01	0.0002913	23.348776	6803.01	0.0002913	23.348776	
	StdDev	2174.9259	9.307E-05	0.0158462	2174.9259	9.307E-05	0.0158462	2174.9259	9.307E-05	0.0158462	
	n	2765.7857	2761.8802	0.0124638	3926.4391	3920.8946	0.0176942	6782.3225	6772.7453	0.0305641	
4 Nodes	Mean	6982.76	0.000299	23.35046	6982.76	0.000299	23.35046	6982.76	0.000299	23.35046	
	StdDev	1940.9112	8.305E-05	0.0127803	1940.9112	8.305E-05	0.0127803	1940.9112	8.305E-05	0.0127803	
	n	2090.6864	2087.8088	0.0081064	2968.0365	2963.9513	0.0115082	5126.8287	5119.7723	0.0198786	
8 Nodes	Mean	7155.76	0.0003064	23.351819	7155.76	0.0003064	23.351819	7155.76	0.0003064	23.351819	
	StdDev	1724.9481	7.381E-05	0.0096884	1724.9481	7.381E-05	0.0096884	1724.9481	7.381E-05	0.0096884	
	n	1572.4336	1570.3231	0.004658	2232.3005	2229.3043	0.0066127	3855.9575	3850.782	0.0114224	
16 Nodes	Mean	7338.73	0.0003144	23.335646	7338.73	0.0003144	23.335646	7338.73	0.0003144	23.335646	
	StdDev	1522.5517	6.512E-05	0.0183914	1522.5517	6.512E-05	0.0183914	1522.5517	6.512E-05	0.0183914	
	n	1164.7537	1160.6092	0.0168082	1653.539	1647.6553	0.0238617	2856.2355	2846.0723	0.0412174	
32 Nodes	Mean	6906.17	0.0002961	23.315799	6906.17	0.0002961	23.315799	6906.17	0.0002961	23.315799	
	StdDev	1883.7584	8.059E-05	0.0376733	1883.7584	8.059E-05	0.0376733	1883.7584	8.059E-05	0.0376733	
	n	2013.2963	2004.6624	0.0706477	2858.1698	2845.9128	0.1002948	4937.0509	4915.8788	0.1732439	
64 Nodes	Mean	6976.43	0.0002992	23.301856	6976.43	0.0002992	23.301856	6976.43	0.0002992	23.301856	
	StdDev	1998.8316	8.557E-05	0.0578138	1998.8316	8.557E-05	0.0578138	1998.8316	8.557E-05	0.0578138	
	n	2221.3539	2212.8619	0.1665767	3153.5382	3141.4826	0.2364801	5447.2545	5426.4304	0.4084832	
128 Nodes	Mean	6858.7	0.0002944	23.276747	6858.7	0.0002944	23.276747	6858.7	0.0002944	23.276747	
	StdDev	2100.0504	8.982E-05	0.084446	2100.0504	8.982E-05	0.084446	2100.0504	8.982E-05	0.084446	
	n	2536.9251	2519.3609	0.3561602	3601.5378	3576.603	0.5056217	6221.1053	6178.0342	0.8733841	

Figure A-16 RC Opt1 Number of Independent Replications Statistics

## A.4.3 VR Basic

n at 1% with 100 samples w/ z = 1.645				n at 1% with 100 samples w/ z = 1.96				n at 1% with 100 samples w/ z = 2.576			
	Worm Size	Latency	Throughput	Worm Size	Latency	Throughput		Worm Size	Latency	Throughput	
2 Nodes	Mean	6982.16	0.000299	23.349854	6982.16	0.000299	23.349854	6982.16	0.000299	23.349854	
	StdDev	2043.8819	8.746E-05	0.0146193	2043.8819	8.746E-05	0.0146193	2043.8819	8.746E-05	0.0146193	
	n	2318.8027	2315.6124	0.0106076	3291.8811	3287.352	0.015059	5686.2207	5678.3975	0.0260121	
4 Nodes	Mean	6920.97	0.0002964	23.349805	6920.97	0.0002964	23.349805	6920.97	0.0002964	23.349805	
	StdDev	2016.9752	8.631E-05	0.0145651	2016.9752	8.631E-05	0.0145651	2016.9752	8.631E-05	0.0145651	
	n	2298.2591	2295.0682	0.0105291	3262.7164	3258.1865	0.0149475	5635.8432	5628.0185	0.0258196	
8 Nodes	Mean	7169.55	0.000307	23.351883	7169.55	0.000307	23.351883	7169.55	0.000307	23.351883	
	StdDev	1723.134	7.373E-05	0.0096143	1723.134	7.373E-05	0.0096143	1723.134	7.373E-05	0.0096143	
	n	1563.0977	1561.003	0.0045869	2219.0468	2216.0731	0.0065118	3833.0636	3827.9271	0.0112481	
16 Nodes	Mean	7221.53	0.0003095	23.332813	7221.53	0.0003095	23.332813	7221.53	0.0003095	23.332813	
	StdDev	1629.2635	6.971E-05	0.0176468	1629.2635	6.971E-05	0.0176468	1629.2635	6.971E-05	0.0176468	
	n	1377.3871	1373.0341	0.0154786	1955.4033	1949.2236	0.0219741	3377.6599	3366.9855	0.037957	
32 Nodes	Mean	6762.92	0.00029	23.31192	6762.92	0.00029	23.31192	6762.92	0.00029	23.31192	
	StdDev	1976.6924	8.457E-05	0.040318	1976.6924	8.457E-05	0.040318	1976.6924	8.457E-05	0.040318	
	n	2311.7534	2301.6332	0.0809419	3281.8735	3267.5064	0.1149089	5668.9342	5644.1171	0.1984875	
64 Nodes	Mean	6876.63	0.000295	23.300139	6876.63	0.000295	23.300139	6876.63	0.000295	23.300139	
	StdDev	2090.5317	8.948E-05	0.0548376	2090.5317	8.948E-05	0.0548376	2090.5317	8.948E-05	0.0548376	
	n	2500.8866	2490.1927	0.1498898	3550.3759	3535.1944	0.2127905	6132.731	6106.5072	0.3675631	
128 Nodes	Mean	6886.09	0.0002955	23.279542	6886.09	0.0002955	23.279542	6886.09	0.0002955	23.279542	
	StdDev	2127.8104	9.1E-05	0.089228	2127.8104	9.1E-05	0.089228	2127.8104	9.1E-05	0.089228	
	n	2583.7606	2565.9593	0.397544	3668.0278	3642.7562	0.5643722	6335.9565	6292.3037	0.9748665	

Figure A-17 VR Basic Number of Independent Replications Statistics

## A.4.4 VR Opt2

n at 1% with 100 samples w/ z = 1.645				n at 1% with 100 samples w/ z = 1.96				n at 1% with 100 samples w/ z = 2.576			
	Worm Size	Latency	Throughput		Worm Size	Latency	Throughput		Worm Size	Latency	Throughput
2 Nodes	Mean	7055.49	0.00030212	23.349948	7055.49	0.00030212	23.349948	7055.49	0.00030212	23.349948	
	StanDev	2026.26152	8.6705E-05	0.01546534	2026.26152	8.6705E-05	0.01546534	2026.26152	8.6705E-05	0.01546534	
	n	2231.86766	2228.82803	0.01187078	3168.464	3164.1488	0.01685231	5473.0366	5465.58274	0.02910979	
4 Nodes	Mean	6853.32	0.00029346	23.348859	6853.32	0.00029346	23.348859	6853.32	0.00029346	23.348859	
	StanDev	2111.09	9.0335E-05	0.01671596	2111.09	9.0335E-05	0.01671596	2111.09	9.0335E-05	0.01671596	
	n	2567.69408	2564.0916	0.01386958	3645.21894	3640.10469	0.01968991	6296.55778	6287.72369	0.0340113	
8 Nodes	Mean	7178.1	0.00030736	23.351734	7178.1	0.00030736	23.351734	7178.1	0.00030736	23.351734	
	StanDev	1748.5153	7.482E-05	0.01010557	1748.5153	7.482E-05	0.01010557	1748.5153	7.482E-05	0.01010557	
	n	1605.65291	1603.50344	0.00506775	2279.46018	2276.40868	0.00719441	3937.41856	3932.14757	0.01242724	
16 Nodes	Mean	7247.17	0.00031056	23.33315	7247.17	0.00031056	23.33315	7247.17	0.00031056	23.33315	
	StanDev	1556.66887	6.6598E-05	0.01685337	1556.66887	6.6598E-05	0.01685337	1556.66887	6.6598E-05	0.01685337	
	n	1248.49666	1244.43117	0.01411752	1772.42442	1766.65285	0.0200419	3061.59189	3051.6224	0.03461931	
32 Nodes	Mean	7012.74	0.00030067	23.316547	7012.74	0.00030067	23.316547	7012.74	0.00030067	23.316547	
	StanDev	1767.13304	7.5574E-05	0.03409809	1767.13304	7.5574E-05	0.03409809	1767.13304	7.5574E-05	0.03409809	
	n	1718.28352	1709.58134	0.05781132	2439.35586	2422.00185	0.08215686	4213.61389	4192.27421	0.1419134	
64 Nodes	Mean	6847.79	0.00029109	23.296182	6847.79	0.00029109	23.296182	6847.79	0.00029109	23.296182	
	StanDev	2075.28448	9.0903E-05	0.0630205	2075.28448	9.0903E-05	0.0630205	2075.28448	9.0903E-05	0.0630205	
	n	2485.34232	2639.01711	0.19802774	3528.30852	3746.47245	0.28112946	6094.61292	6471.45773	0.48560812	
128 Nodes	Mean	6804.34	0.00029202	23.279424	6804.34	0.00029202	23.279424	6804.34	0.00029202	23.279424	
	StanDev	2161.37281	9.2441E-05	0.08485597	2161.37281	9.2441E-05	0.08485597	2161.37281	9.2441E-05	0.08485597	
	n	2730.35527	2711.61139	0.35954412	3876.14039	3849.53071	0.5104257	6695.43923	6649.47508	0.88168227	

Figure A-18 VR Opt2 Number of Independent Replications Statistics

## A.4.5 VR Opt4

n at 1% with 100 samples w/ z = 1.645				n at 1% with 100 samples w/ z = 1.96				n at 1% with 100 samples w/ z = 2.576			
	Worm Size	Latency	Throughput		Worm Size	Latency	Throughput		Worm Size	Latency	Throughput
2 Nodes	Mean	6938.37	0.0002971	23.349598	6938.37	0.0002971	23.349598	6938.37	0.0002971	23.349598	
	StanDev	2088.1304	8.935E-05	0.0147746	2088.1304	8.935E-05	0.0147746	2088.1304	8.935E-05	0.0147746	
	n	2450.937	2447.5434	0.0108344	3479.4651	3474.6474	0.015381	6010.2434	6001.9215	0.0265682	
4 Nodes	Mean	7190.58	0.0003079	23.35201	7190.58	0.0003079	23.35201	7190.58	0.0003079	23.35201	
	StanDev	1696.3258	7.259E-05	0.0091651	1696.3258	7.259E-05	0.0091651	1696.3258	7.259E-05	0.0091651	
	n	1505.9915	1503.9793	0.0041682	2137.9761	2135.1195	0.0059174	3693.0265	3688.0922	0.0102215	
8 Nodes	Mean	7240.33	0.00031	23.351995	7240.33	0.00031	23.351995	7240.33	0.00031	23.351995	
	StanDev	1693.4708	7.246E-05	0.0097124	1693.4708	7.246E-05	0.0097124	1693.4708	7.246E-05	0.0097124	
	n	1480.3707	1478.4076	0.004681	2101.6037	2098.8167	0.0066454	3630.1988	3625.3847	0.0114788	
16 Nodes	Mean	7130.32	0.0003057	23.310601	7130.32	0.0003057	23.310601	7130.32	0.0003057	23.310601	
	StanDev	1795.9731	7.661E-05	0.0200385	1795.9731	7.661E-05	0.0200385	1795.9731	7.661E-05	0.0200385	
	n	1716.7751	1699.2397	1.9996566	2437.2145	2412.3204	2.8388063	4209.915	4166.9142	4.9036034	
32 Nodes	Mean	6675.57	0.0002863	23.309234	6675.57	0.0002863	23.309234	6675.57	0.0002863	23.309234	
	StanDev	2031.0385	8.688E-05	0.0444911	2031.0385	8.688E-05	0.0444911	2031.0385	8.688E-05	0.0444911	
	n	2504.9059	2492.4895	0.0985875	3556.0819	3538.455	0.1399594	6142.5872	6112.1394	0.2417585	
64 Nodes	Mean	6911.51	0.0002965	23.30268	6911.51	0.0002965	23.30268	6911.51	0.0002965	23.30268	
	StanDev	1956.6561	8.372E-05	0.0464796	1956.6561	8.372E-05	0.0464796	1956.6561	8.372E-05	0.0464796	
	n	2168.777	2157.884	0.1076576	3078.8976	3063.4333	0.1528357	5318.3243	5291.6121	0.2640004	
128 Nodes	Mean	6976.31	0.0002994	23.282664	6976.31	0.0002994	23.282664	6976.31	0.0002994	23.282664	
	StanDev	1982.6967	8.479E-05	0.0719573	1982.6967	8.479E-05	0.0719573	1982.6967	8.479E-05	0.0719573	
	n	2185.7116	2170.3081	0.2584733	3102.9386	3081.0712	0.3669409	5359.8515	5322.0788	0.6338342	

Figure A-19 VR Opt4 Number of Independent Replications Statistics



## Appendix B: Designer Block Diagrams

The figures presented in this appendix provide the logic diagrams of the major blocks created for this research project. The following subsections are arranged to show the highest-level view of a component followed by the sub-components. Each sub-component will be followed by a display of the corresponding sub-blocks within it as needed. Section B.1 shows the switch and the sub-components within it. The next section, B.2, shows the two different workstation views, *Form* and *Stat* and Section B.3 displays the topologies used during the simulation tests. Finally, B.4 provides a view of the model validation tests.

### B.1 Switch

The first block diagram inside the switch shows the complexity of combining the logic to produce an 8-port switch. This view shows the integration of various blocks that

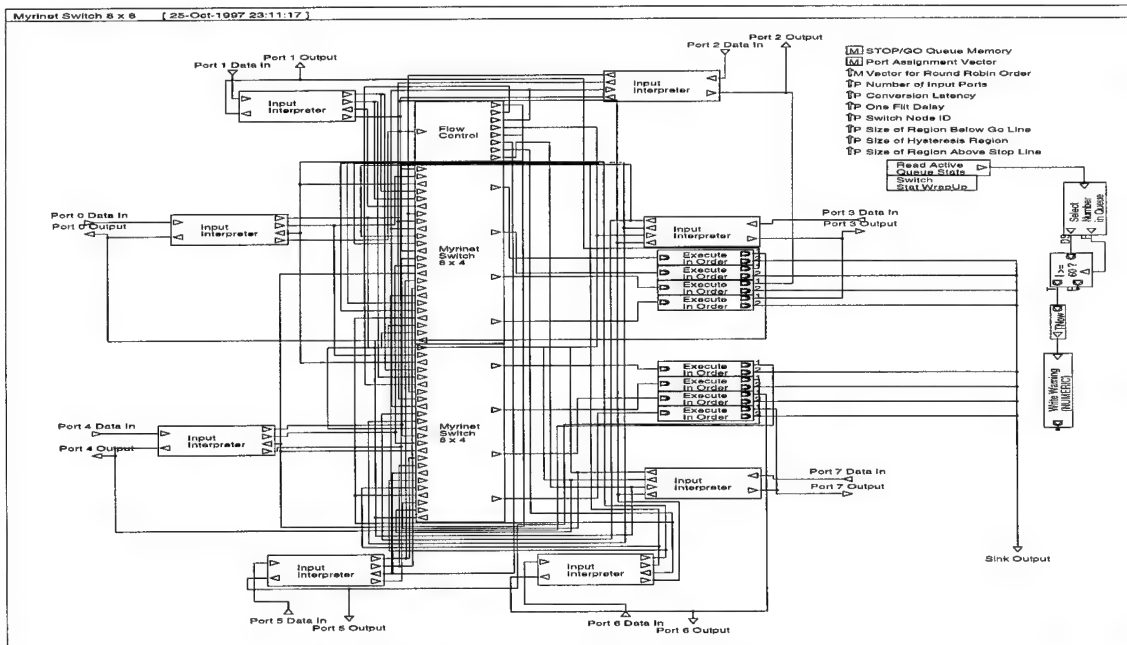


Figure B-1 Myrinet 8-Port Switch Block Diagram



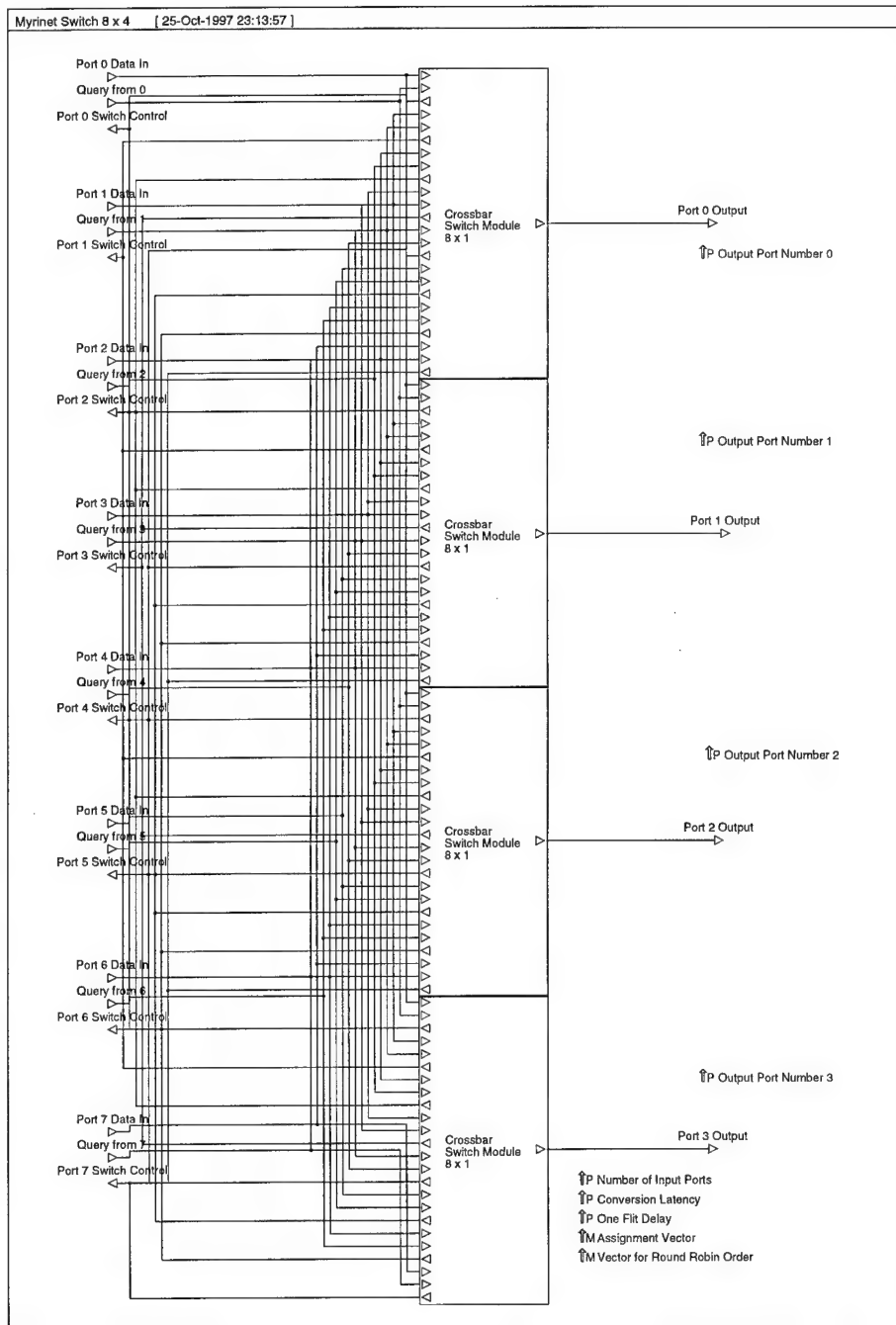


Figure B-3 Myrinet Switch 8x4 Block Diagram

### B.1.3 Crossbar Switch Module 8x1

The Crossbar Switch Module 8 x 1 shows the combination of the arbitration logic and the cross point functionality. Placing these blocks together requires the number of

input ports to be known. Any increase beyond eight inputs requires the additional logic to be added.

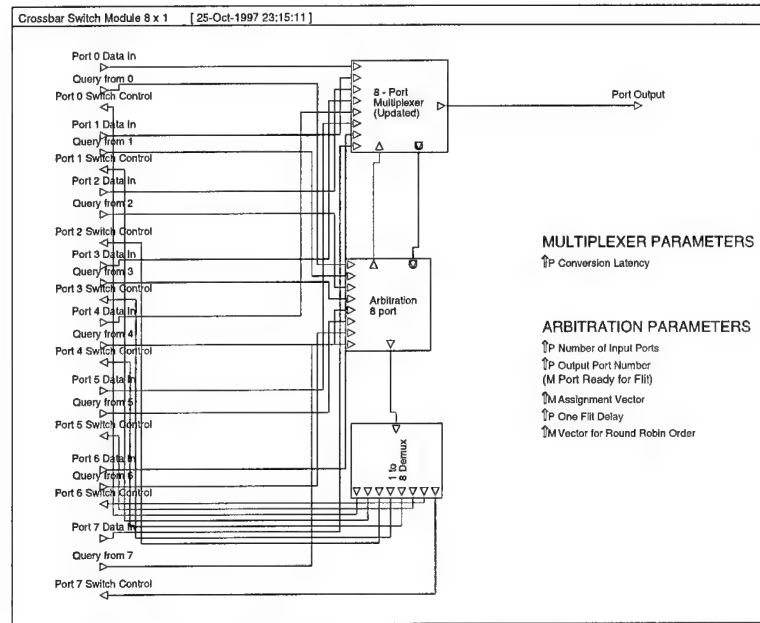


Figure B-4 Crossbar Switch Module 8 x 1 Block Diagram

#### B.1.4 8-Port Multiplexer

The 8-Port Multiplexor accepts a flit as an input and checks for the cross-point to be set. If the input port's cross-point is set, then the flit is forwarded.

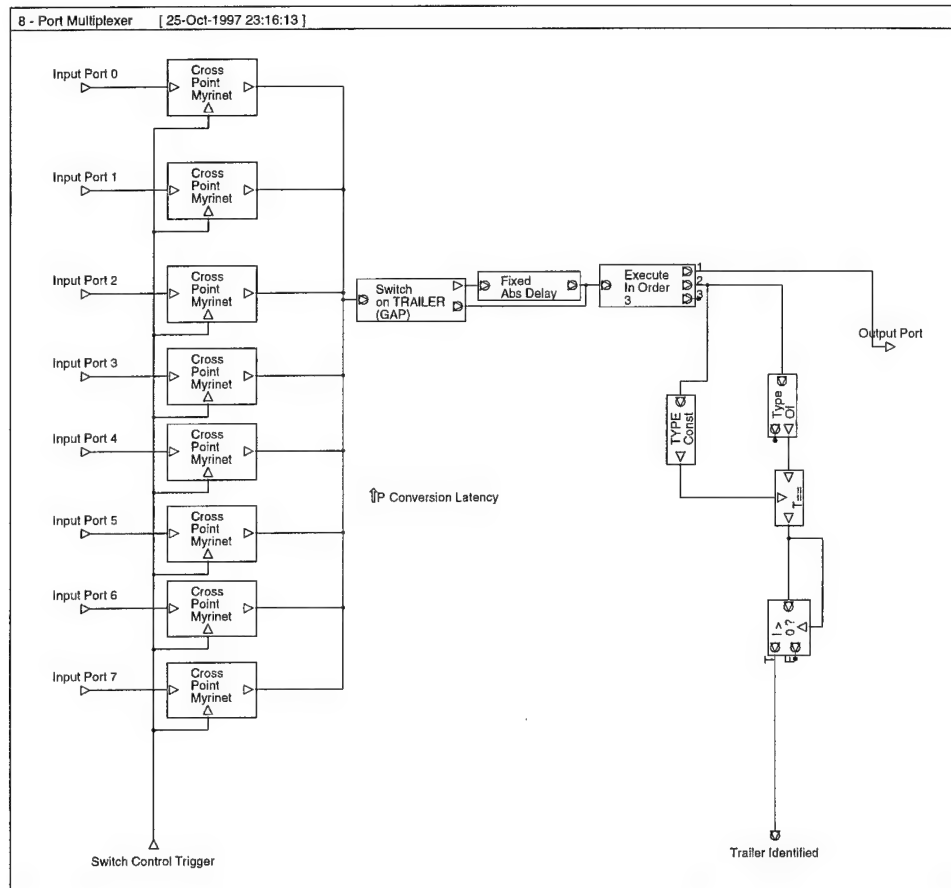


Figure B-5 8-Port Multiplexor Block Diagram

### B.1.5 Cross Point Myrinet

The Cross Point Myrinet block is responsible for setting a cross point or releasing it in response to the arbitration's input. In addition, the block generates a new GAP symbol in response to a new setting.

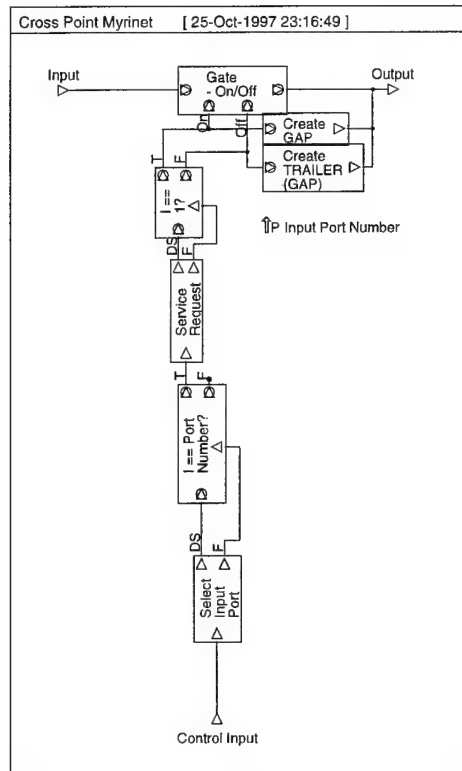


Figure B-6 Cross Point Myrinet

#### B.1.6 Arbitration 8-Port

The Arbitration is responsible for maintaining the connections to a given output port. The arbitrator tracks who is connected, who has requested, and who is next in line for arbitration assignment by using the Pick Winner block.

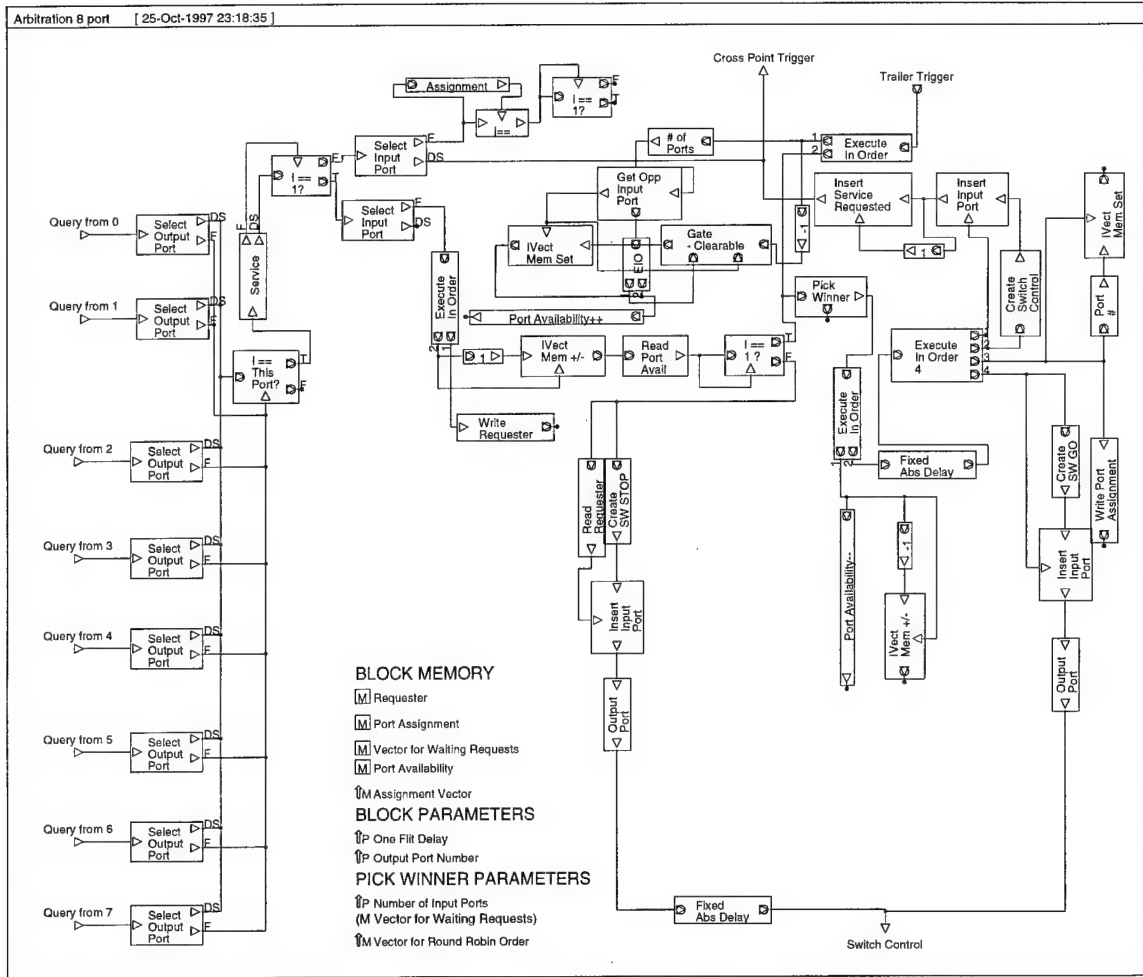


Figure B-7 Arbitration 8-Port Block Diagram

### B.1.7 Pick Winner

The Pick Winner module is responsible for using the arbitration order supplied at execution time to maintain the next input port to which the output port will be awarded. This module is a sub-component of the Arbitration process.

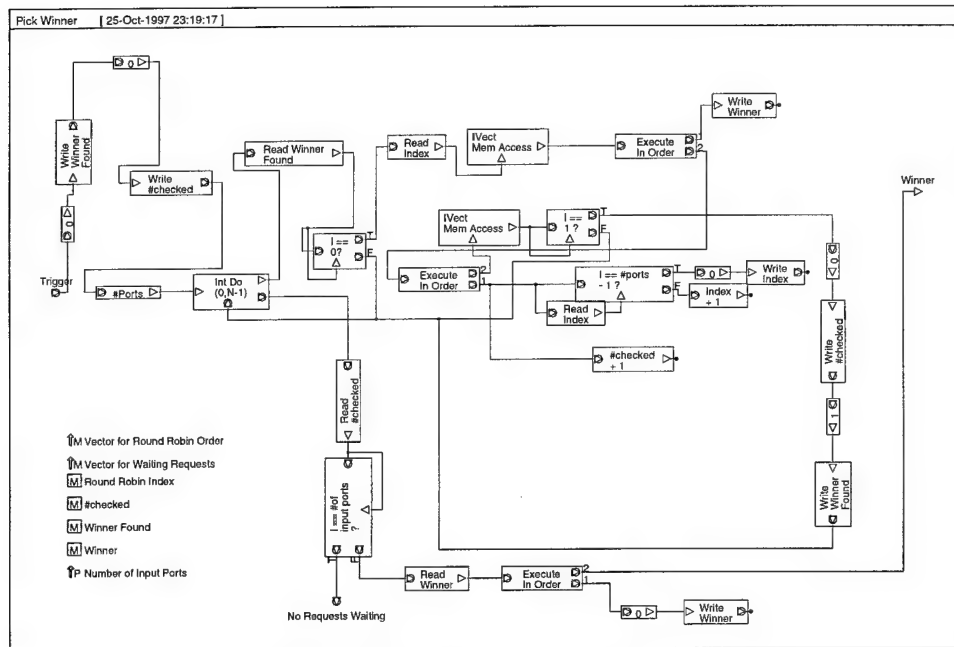


Figure B-8 Pick Winner Block Diagram

## B.2 Workstations

There are two types of workstations: a form workstation and a stat workstation. The form workstation is used when the captured network traffic is used as input to the model. In contrast, the stat workstation is used when a statistical representation of the network dynamics is emulated.

### B.2.1 Form Workstation

As stated above, a form workstation is used when the simulation emulates a Myrinet with captured traffic as inputs. By including a special source process, destination process, and Myrinet adapter process, the Form Workstation was developed which provided a “workstation” view of the necessary functions.



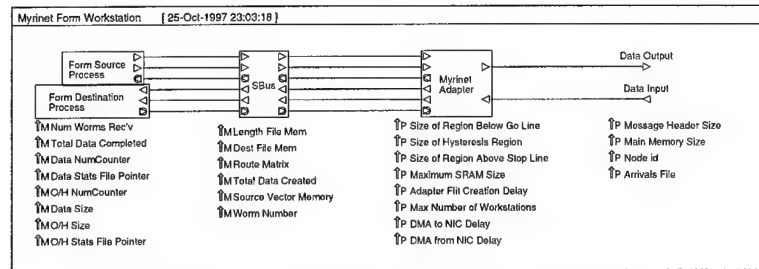


Figure B-9 Myrinet Form Workstation Block Diagram

### B.2.2 Form Source Process

The Form Source Process block provides the necessary logic to read the traffic distribution stream from a file, to identify the necessary owner of the next worm, to assign appropriate delivery data, and to send a worm to a Myrinet adapter module.

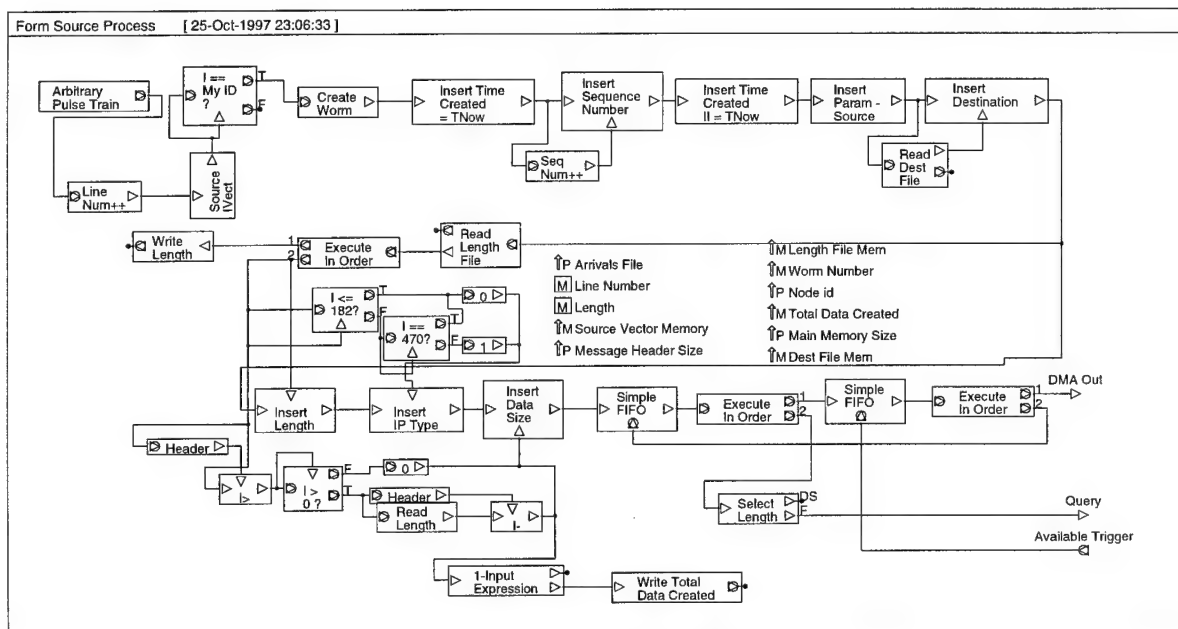


Figure B-10 Form Source Process Block Diagram

### B.2.3 Myrinet Adapter

The Myrinet Adapter module controls the sending of a worm across a Myrinet. The module reads a worm's destination, assigns the appropriate route, and flitizes a worm

for the sending process. The sending process is composed of creating a GAP symbol, sending it and the remaining route and data flits across the network, and responding to control symbols as necessary.

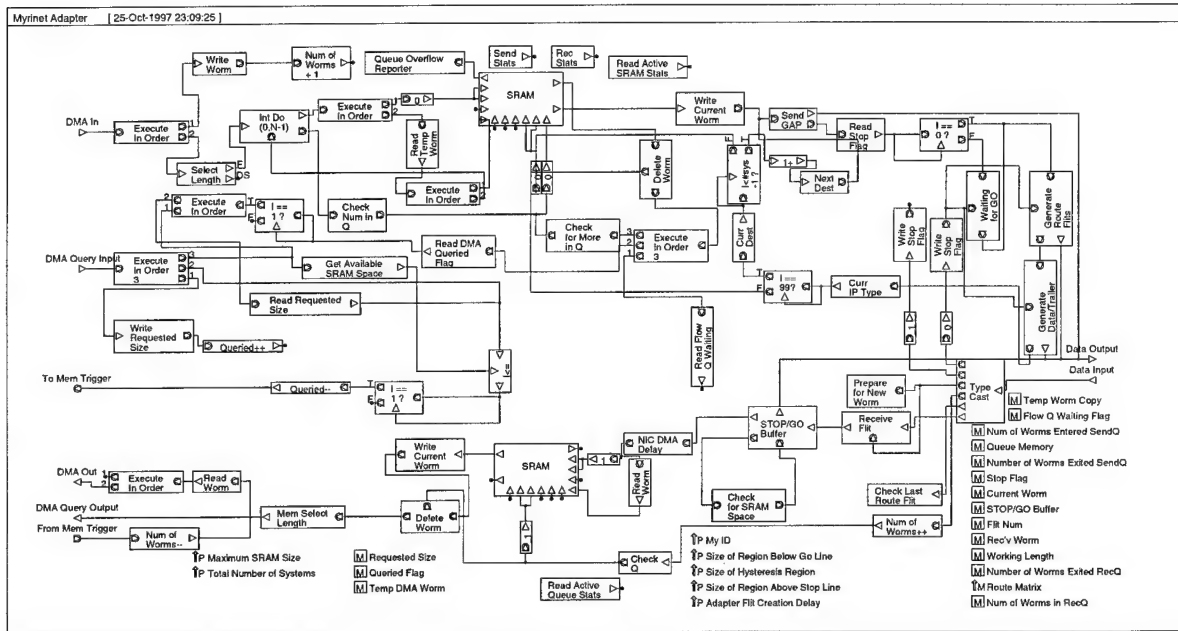


Figure B-11 Myrinet Adapter Block Diagram

#### B.2.4 Generate Route Flits

The Generate Route Flits module produces the necessary flits needed to move a worm through a Myrinet. The flits generated by this process are unbounded in number and are capable of navigating arbitrarily connected nodes. The only requirement is that the nodes be valid Myrinet compliant devices.

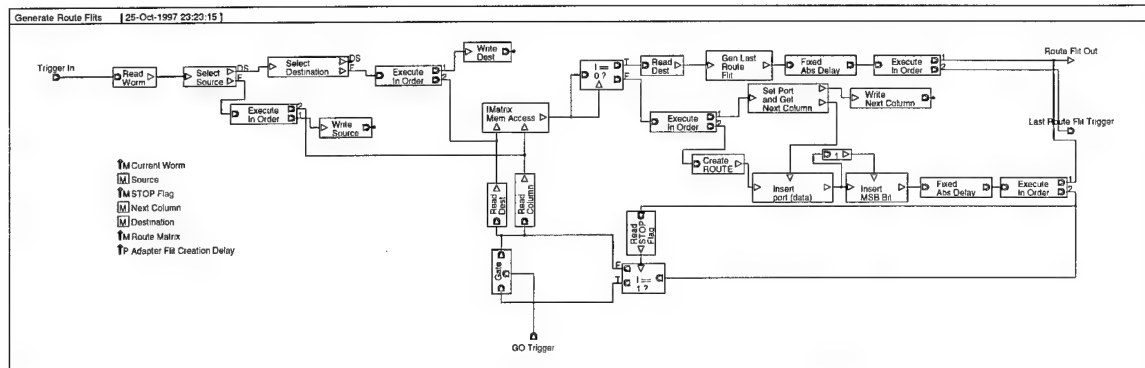


Figure B-12 Generate Route Flit Block Diagram

### B.2.5 Stat Workstation

A Stat Workstation module is used when the simulation tests a Myrinet with statistically emulated traffic patterns. The corresponding modules within this block are a special source process, a destination process, and the previously displayed Myrinet Adapter module.

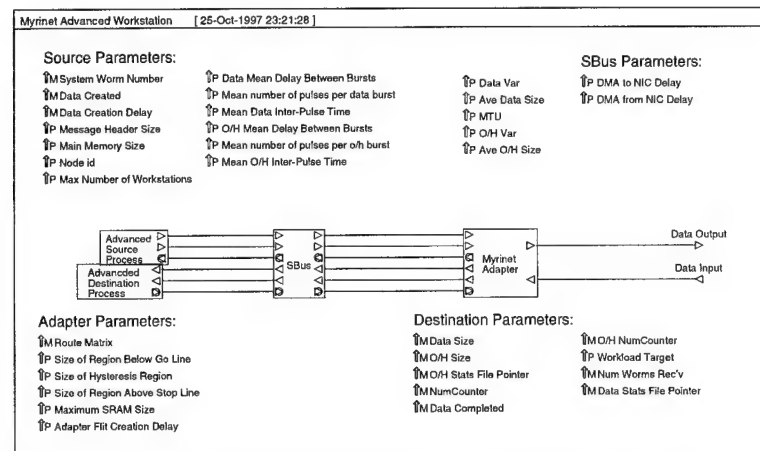


Figure B-13 Myrinet Advanced (Stat) Workstation Block Diagram

### B.2.6 Advanced Source Process

The Advanced Source Process module provides the necessary logic to implement statistically represented traffic generation. Using BoNES's built in primitives, the block creates a worm based on a given window time and inter-arrival time within that window.

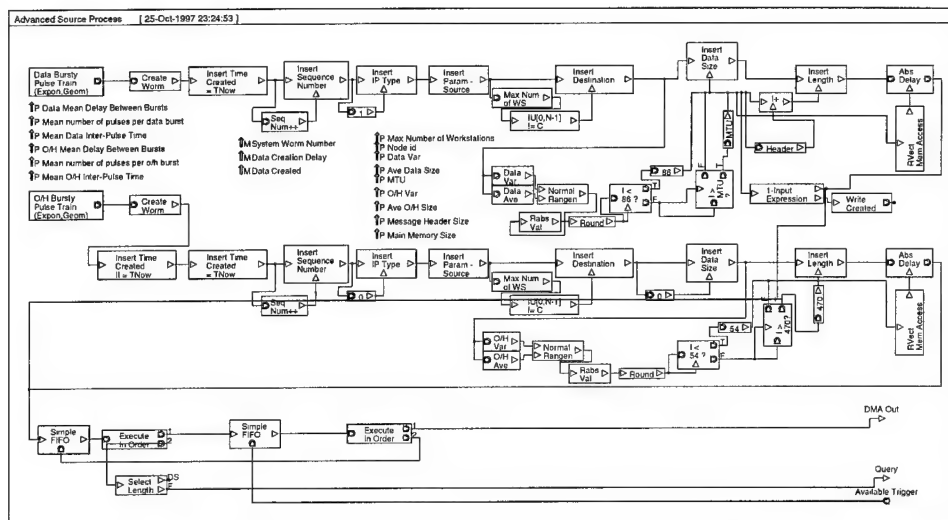


Figure B-14 Advanced Source Process Block Diagram

### B.2.7 Advanced Destination Process

The Advanced Destination Process is responsible for tracking the completed data worms and completed overhead worms. The statistics kept on these two items is shared between all workstations within the network.

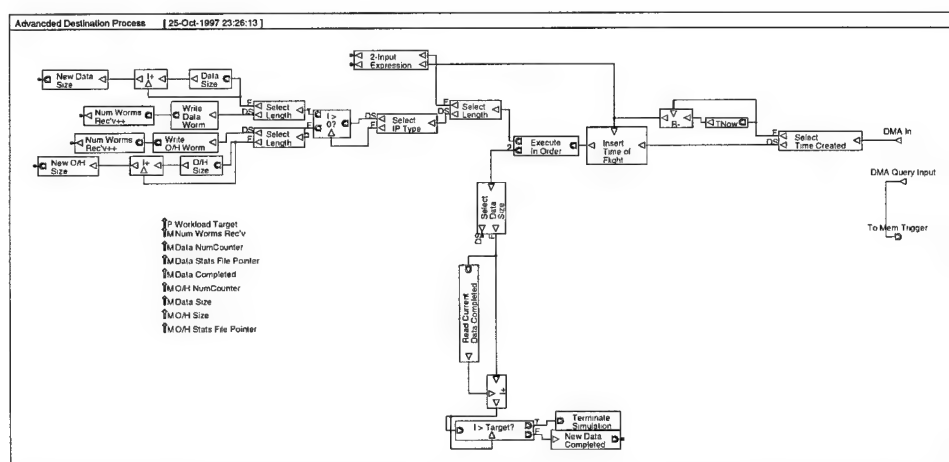


Figure B-15 Advanced Destination Process Block Diagram

### B.3 Simulation Topologies

The simulation topologies consist of the necessary block(s) to integrate the previously listed blocks to build high-order configurations. The blocks range from 2-node topologies to the advanced 128-node topologies where the term node as used is a workstation. In reality, the actual number of Myrinet nodes is higher due to the number of switches used in a given topology.

#### B.3.1 2-Nodes Trace Topology

This topology was used for the following simulations: PCI\_33; PCI\_66; RCB\_Trace\_One\_File; RCB\_Trace\_2; RC1\_Trace\_2; VRB\_Trace\_2; VR2\_Trace\_2; and VR4\_Trace\_2.

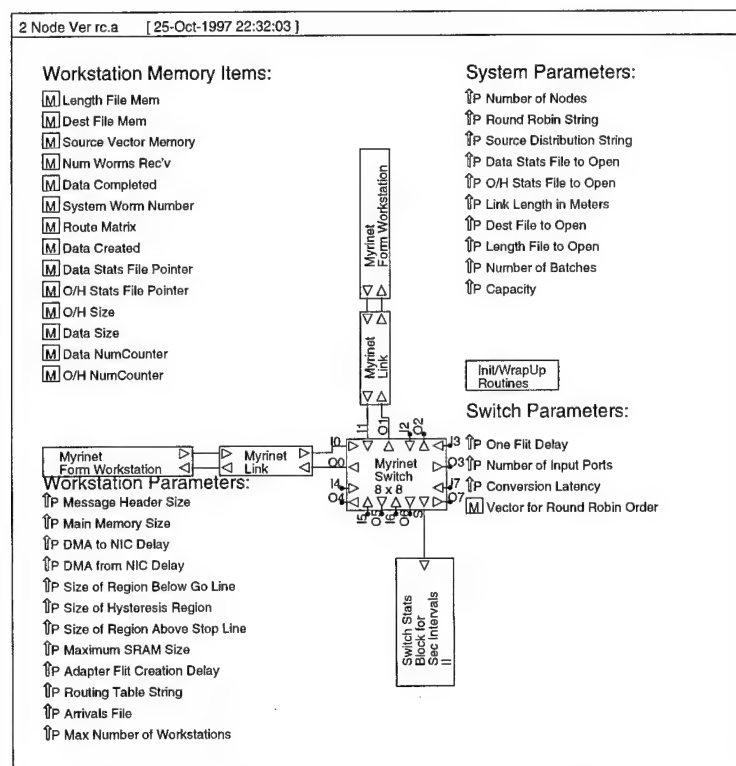


Figure B-16 2-Node Trace Topology Block Diagram

### B.3.2 4-Nodes Trace Topology

This topology was used for the following simulations: RCB\_Trace\_4; RC1\_Trace\_4; VRB\_Trace\_4; VR2\_Trace\_4; and VR4\_Trace\_4.

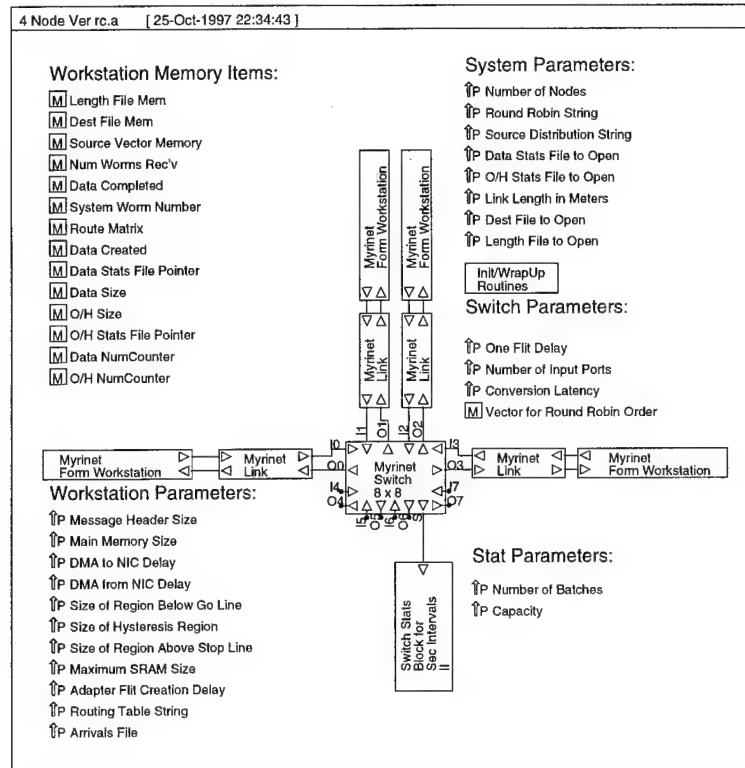


Figure B-17 4-Node Trace Topology Block Diagram

### B.3.3 2-Nodes Stat Topology

This topology was used for the following simulations: RCB\_Stat\_2; RC1\_Stat\_2; VRB\_Stat\_2; VR2\_Stat\_2; and VR4\_Stat\_2. This is the first topology where the traffic dynamics are based on the statistical representation of the algorithms.

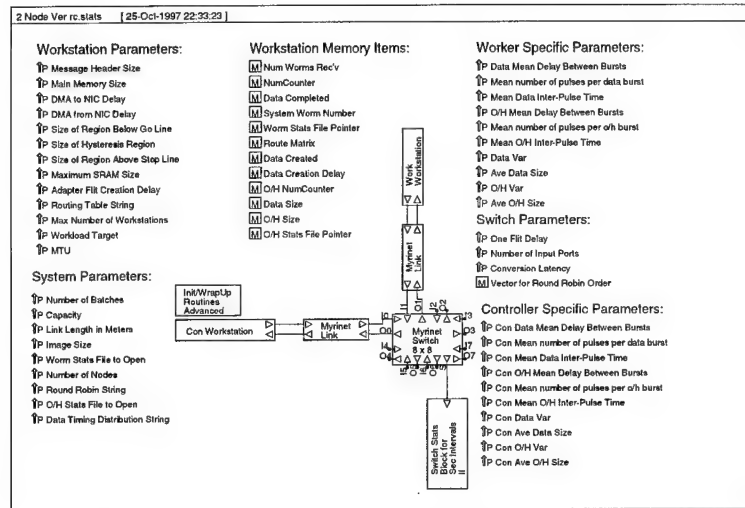


Figure B-18 2-Node Stat Topology Block Diagram

### B.3.4 4-Nodes Stat Topology

This topology was used for the following simulations: RCB\_Stat\_4; RC1\_Stat\_4; VRB\_Stat\_4; VR2\_Stat\_4; and VR4\_Stat\_4.

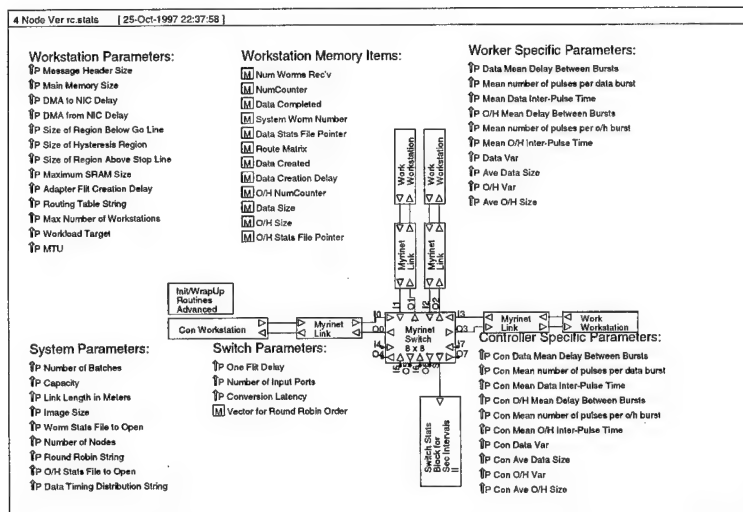


Figure B-19 4-Node Stat Topology Block Diagram

### B.3.5 8-Nodes Stat Topology

This topology was used for the following simulations: RCB\_Stat\_8; RC1\_Stat\_8; VRB\_Stat\_8; VR2\_Stat\_8; and VR4\_Stat\_8. This is the first simulation for which no corresponding actual system configuration could be tested in-house. In addition, this is the last topology for which all nodes are accessible by traversing only two links.

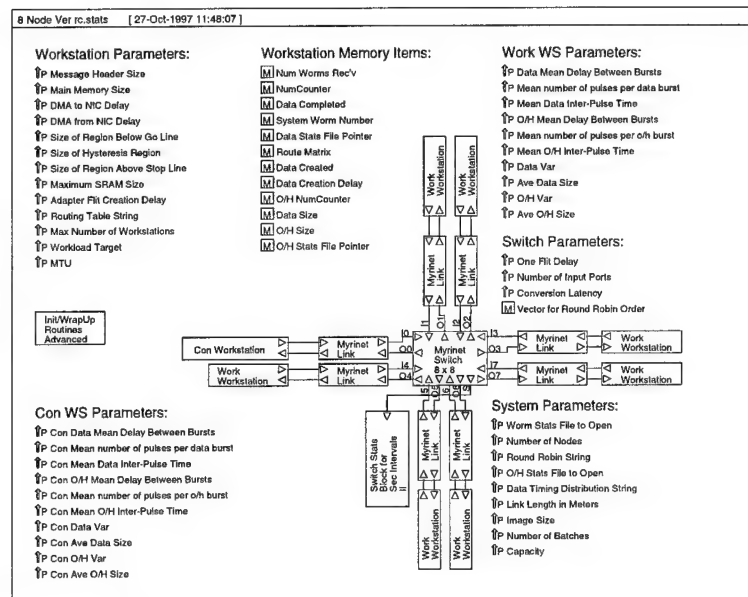


Figure B-20 8-Node Stat Topology Block Diagram

### B.3.6 16-Nodes Stat Topology

This topology was used for the following simulations: RCB\_Stat\_16; RC1\_Stat\_16; VRB\_Stat\_16; VR2\_Stat\_16; and VR4\_Stat\_16.



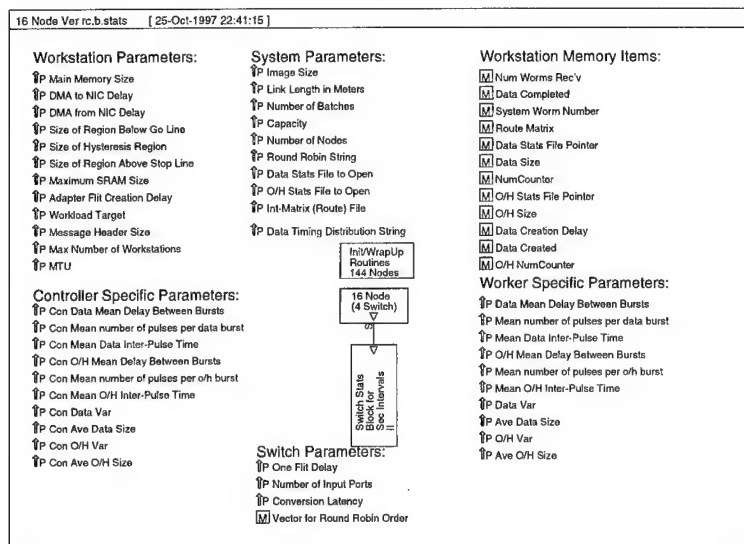


Figure B-21 16-Node Stat Topology Block Diagram

### B.3.7 Init/Wrap-Up Routines 144-Nodes

The initialization and end-of-simulation routines are necessary to coordinate the overhead activities of a simulation. Functions included within the Init/Wrap-Up Routines 144-Nodes module are portable between the 16, 32, 64, and 128-nodes topologies.

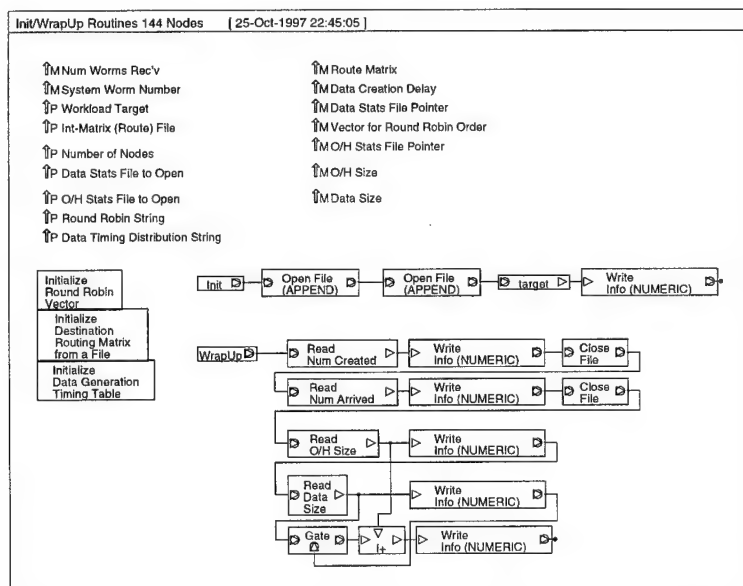


Figure B-22 Init/Wrap-Up Routines 144-Nodes Block Diagram

### B.3.8 16-Nodes (4 Switches)

The 16-Nodes (4 Switches) module is the nested view of the network within the 16-Nodes Stat Topology block. This block allows the next lower level, the 16-Nodes (3 Switches) module, to be connected via a “fat tree” structure.

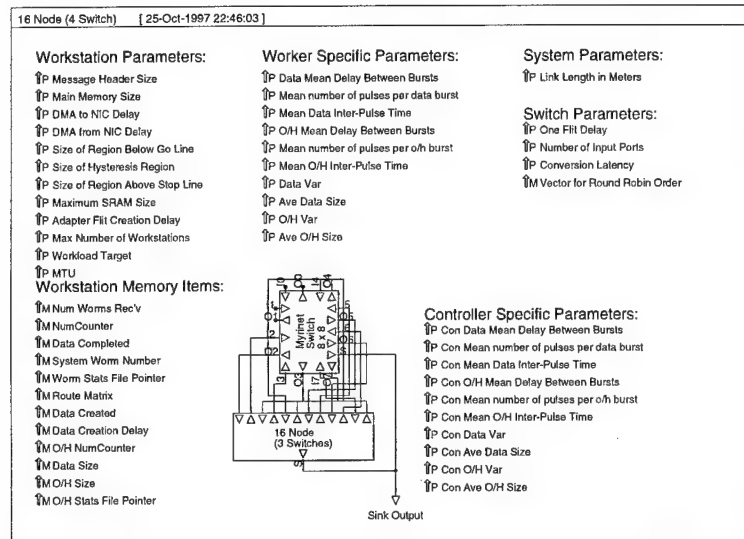


Figure B-23 16-Nodes (4 Switches) Block Diagram

### B.3.9 16-Nodes (3 Switches)

The 16-Nodes (3 Switches) module contains the actual systems used in the 16-nodes stat simulations. This block is incorporated into a higher-level process to ease diagramming.

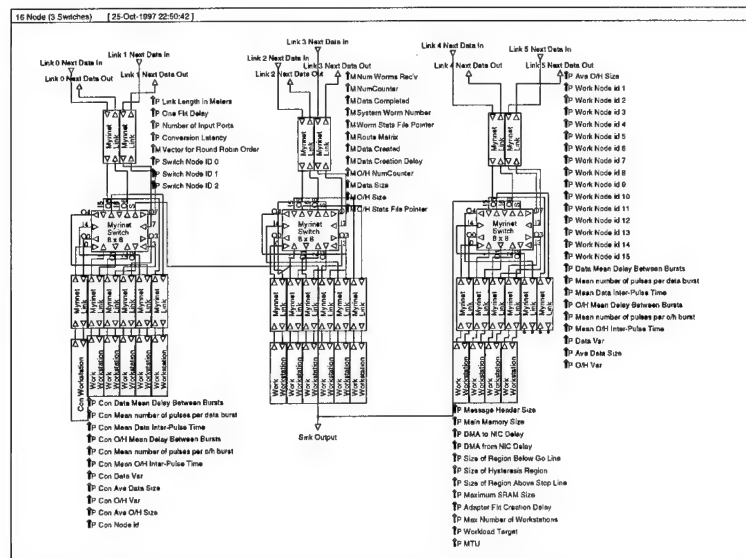
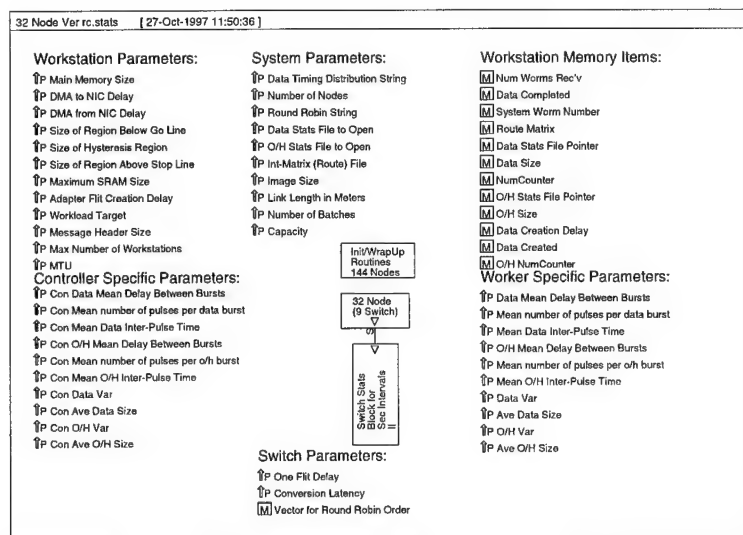


Figure B-24 16-Nodes (3 Switches)

### B.3.10 32-Nodes Stat Topology

This topology was used for the following simulations: RCB\_Stat\_32; RC1\_Stat\_32; VRB\_Stat\_32; VR2\_Stat\_32; and VR4\_Stat\_32.



*Figure B-25 32-Node Stat Topology Block Diagram*

### B.3.11 32-Nodes (9 Switches)

This is the next view within the 32-Nodes Stat Topology where a total of nine switches are used to get the “fat tree” structure completely connected. In turn, a module

consisting of thirty-two workstations and six switches is used to sub-divide the problem of diagramming the topology.

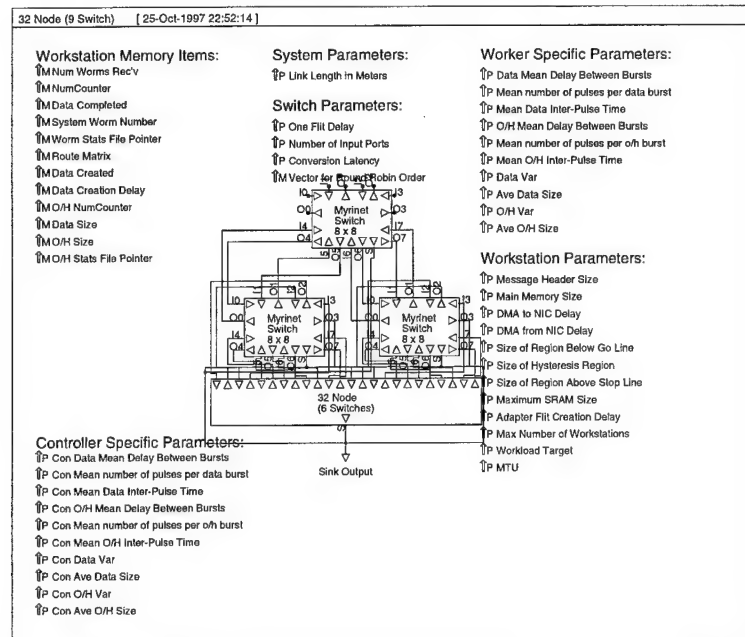


Figure B-26 32-Nodes (9 Switches)

### B.3.12 32-Nodes (6 Switches)

This 32-Nodes (6 Switches) module contains the logic for thirty-two workstations. However, the actual capacity is thirty-six workstations. The lower number is used to represent configurations that are multiples of two resulting from the algorithms tested in this research.



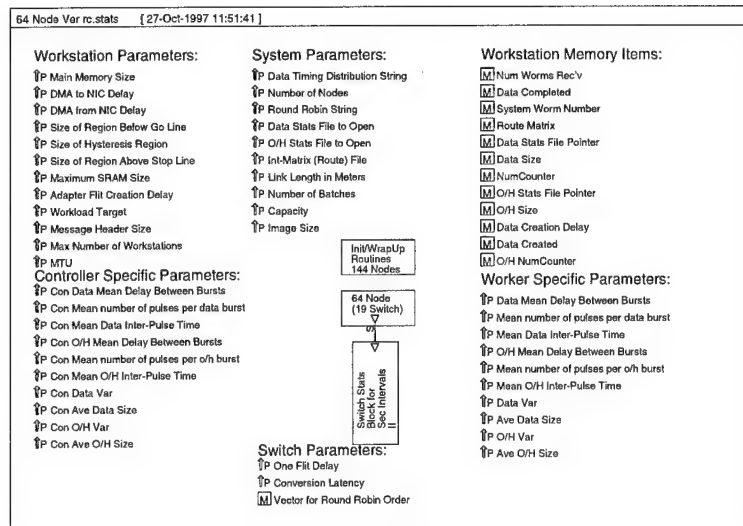


Figure B-28 64-Node Stat Topology Block Diagram

### B.3.14 128-Nodes Stat Topology

This topology was used for the following simulations: RCB\_Stat\_128; RC1\_Stat\_128; VRB\_Stat\_128; VR2\_Stat\_128; and VR4\_Stat\_128.

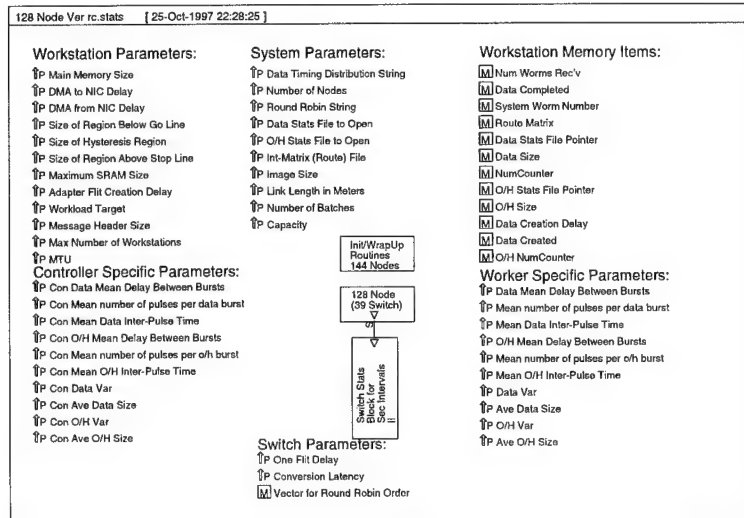


Figure B-29 128-Node Stat Topology Block Diagram

### B.3.15 64-Nodes (19 Switches)

The 64-Nodes (19 Switches) block combines the two thirty-two node modules to create the necessary “fat-tree” topology.

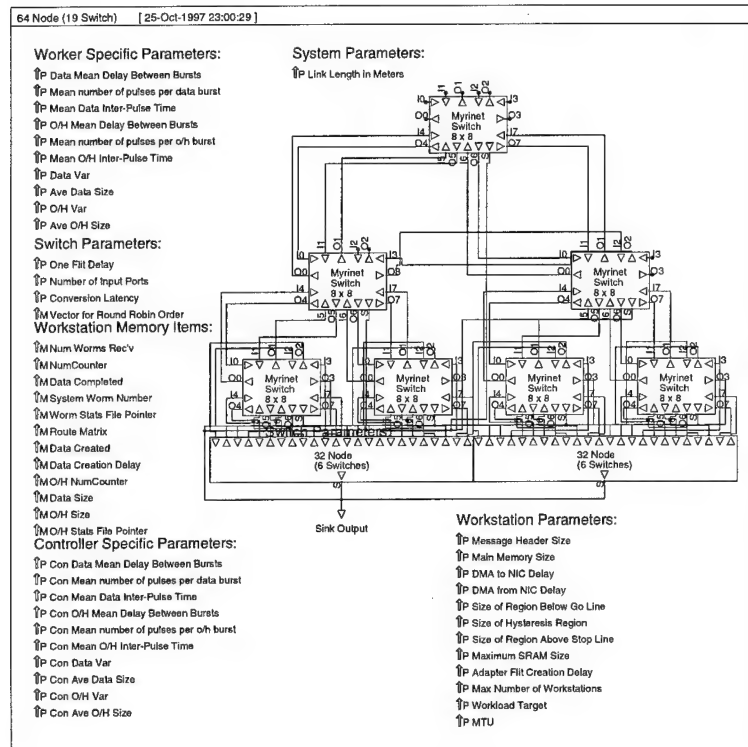


Figure B-30 64-Nodes (19 Switches) Block Diagram

### B.3.16 128-Nodes (39 Switches)

The 128-Nodes (39 Switches) block diagram shows the largest topology created during this research. This block represents the aggregation of four 32-Node (6 Switches) blocks along with fifteen additional switches. The resulting “fat-tree” structure combines the computing power of 128 workstations.

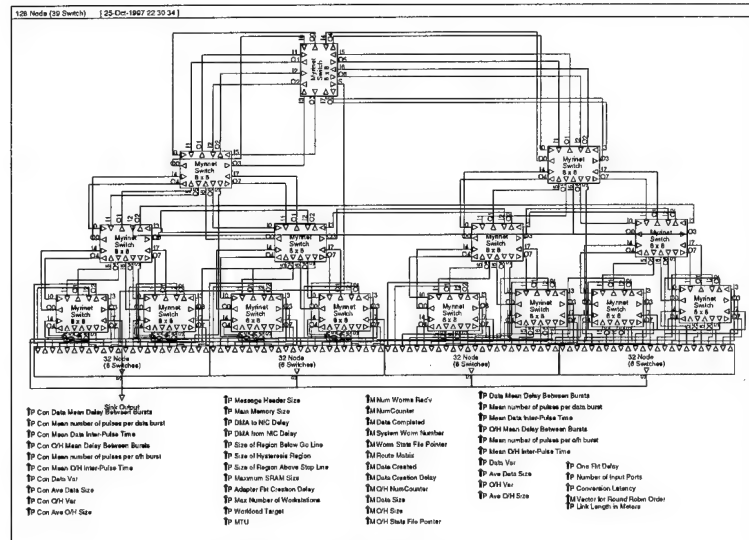


Figure B-31 128-Nodes (39 Switches)

## B.4 Validation Tests

The four validation tests conducted sought to prove the functionality of the model. The first test sought to prove that times achieved by worms, assuming freedom from workstation overhead delay, did not exceed the corresponding maximum theoretical throughput. The second test was a variation on the first with a delay used to emulate the overhead associated with using MPI as the message-passing platform. The third test was similar to the first as it placed a switch between the two systems. Finally, the last test combined a measured overhead for two systems with an inter-connecting switch.

### B.4.1 Validation Test Topology Number One

The first and second validation tests used the same block diagram as the topology of choice. The difference was seen in the input parameters as the first was free from overhead delay. That test compared the theoretical maximum throughput a given worm size could achieve versus the model's throughput. For each measurement, the Sun workstation's parameters were used. This test placed two workstations together with a



single link connecting the two. The second test simply added a delay that was obtained by conducting point-to-point-timing analysis between two systems.

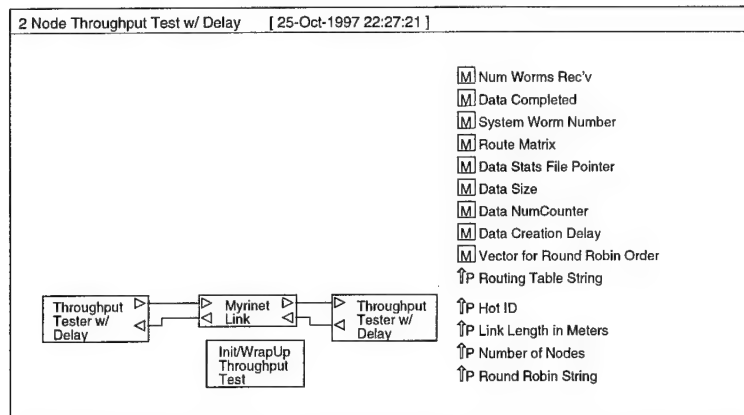


Figure B-32 Two Node Validation Test Block Diagram

#### B.4.2 Validation Test Topology Number Two

The two tests that were conducted using the second topology incurred the additional delay associated with an inter-connecting switch. The remaining parameters were unchanged.

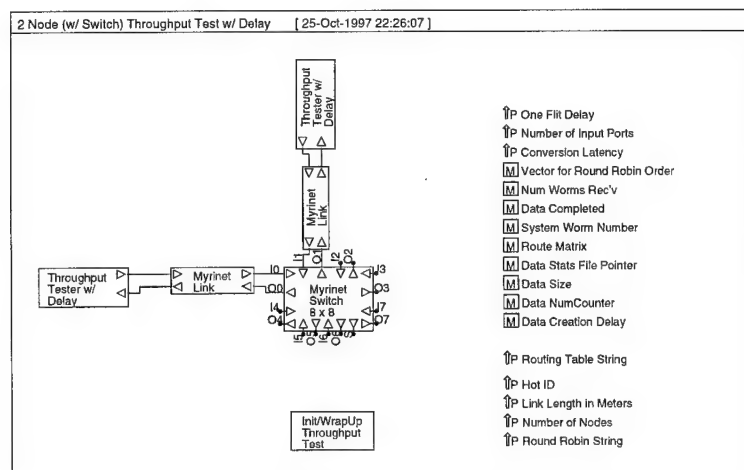


Figure B-33 Two-Node Validation Test Topology Number Two

## **Appendix C: Routing**

### ***C.1 Discussion***

The routing process in the model does not follow Myrinet's standard. A couple of reasons contributed to the diversion. First, Myricom calls the algorithm complex and very foreign to outsiders<sup>10</sup>. The need to reduce the complexity during the initial development and the concept acceptance phases was paramount to the success of this research. Thus, the approach to routing was simplified to assume the networks would know the route and would not perform network mapping. Secondly, the reason for mapping, approximately 6% of network overhead as reported by Myricom, is to create the route table used by nodes for a worm to traverse the network. Moreover, the network route "map" must be tolerant of node initialization/failure. In addition to the model assuming node failures do not occur, it also assumes no new workstations/switches are added after a simulation begins. Therefore, the model does not perform network mapping and the routing table is a required parameter. Section C.2 provides examples of routing tables used as inputs during the model verification stage.

### ***C.2 Examples***

The routing table is a static parameter for all workstations in a given simulation. The process block diagram that is directly responsible for using the table is the Myrinet Adapter and its embedded Generate Route Flits block. The Generate Route Flits (GRF) routine is initiated following the selection of a worm and the creation of a GAP symbol. The first action in the process is to read the current worm's destination field. After determining the destination node, the GRF will read the matrix at the column that

corresponds to the source's node identifier and crossed where the row entry corresponds to the destination's node identifier. This source-destination pair (column, row) will produce an entry that is either a zero or an entry that has at least two digits. The former represents the case where two systems are directly connected such that there are only four entries in the matrix. In all tables, the diagonal will have a "-99" in each entry signifying the fact that a node would try to send to itself which is assumed not to happen. The off entries are 0 telling the two nodes that the requested destination is directly attached. When a "0" is encountered, the GRF generates a "final route flit" that has a one bit high setting that signifies the fact that it is the last one. The "data" field for the final route flit is set to the destination's node identification, which allows the node receiving the worm to check its identification to ensure it was the target of the worm.

The counter case to the previous two-node topology is all topologies consisting of one or more switches. In these instances, the (column, row) pair signifying the (source, destination) pair provides an entry that is at least two digits in length. The right most digit tells the GRF to create a route flit with the "data" field equal to this value. The resulting path taken by the worm will be to exit the port identified in the "data" field at the next switch. In this research, all switches were assumed to have 10 or less ports but, the GRF could be extended to accept a different counting base that would allow larger numbers to be accepted by the right-most digit. The digits to the left of the port designator are the identifier for the next column entry. This identifier is an identification number for a switch within the network. All switches are numbered and given a column

---

<sup>10</sup> Based on discussions with company representatives.

in the route table. The entries in a switches column are either a zero or a value with two or more digits. The GRF takes the identifier for the next column and crosses it with the original destination identifier to create a new (column, row) pair.

Resulting from this newly generated pair is the next value(s) for the GRF to consider when generating the second and successive flits. In each case, if the new pair returns a zero value from the table, then the GRF assumes that the requested destination is connected at that switch (identified by the column value). The resulting “final route flit” contains the one-bit identifier and the node identifier in the flit’s data field. In the case where the table entry returns a non-zero field, the GRF knows that the link from the previous switch is a switch-to-switch connection. Correspondingly, the GRF creates a new route flit with the data field set to the next switch’s port that is needed to traverse the network. Also, the GRF reads the values to the left of the right-most digit to determine the next column to access in the table. Successive entries are read until a “zero” is interpreted that directs the GRF to create the “final route flit.” In this fashion, multiple switch configurations are implemented and arbitrarily laid out. The following subsections provide three examples of routes used during the verification phase.

### ***C.2.1 16-Workstations, 3-Switches Topology***

Figure C-1 gives the table used for a “triangle” shaped topology consisting of three inter-switch links. For this table, a source of “0” and a destination of “10” creates three route flits. The first two are standard route flits with data fields of “5” and “7” respectively. These values result in the worm entering the first switch on port “3” and exiting on port “5” to traverse an inter-switch link. The next action the worm takes is to enter the second switch on port “5” and exit through port “7.” The destination node then

consumes the third, or final route flit, which has "10" in its data field. The three corresponding table entries in this case were: 165, 177, 0.

		Sending Node															Switches			
R e c e i v i n g  N o d e		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
	0	-99	163	163	163	163	175	175	175	175	175	175	186	186	187	187	187	0	160	160
	1	162	-99	162	162	162	175	175	175	175	175	175	186	186	187	187	187	0	161	161
	2	161	161	-99	161	161	175	175	175	175	175	175	186	186	187	187	187	0	162	162
	3	160	160	160	-99	160	175	175	175	175	175	175	186	186	187	187	187	0	163	163
	4	164	164	164	164	-99	175	175	175	175	175	175	186	186	187	187	187	0	164	164
	5	165	165	165	165	165	-99	174	174	174	174	174	185	185	185	185	185	174	0	174
	6	165	165	165	165	165	170	-99	170	170	170	170	185	185	185	185	185	170	0	170
	7	165	165	165	165	165	171	171	-99	171	171	171	185	185	185	185	185	171	0	171
	8	165	165	165	165	165	172	172	172	-99	172	172	185	185	185	185	185	172	0	172
	9	165	165	165	165	165	173	173	173	173	-99	173	185	185	185	185	185	173	0	173
	10	165	165	165	165	165	177	177	177	177	177	-99	185	185	185	185	185	177	0	177
	11	167	167	166	166	166	176	176	176	176	176	176	-99	184	184	184	184	184	184	0
	12	167	167	166	166	166	176	176	176	176	176	176	180	-99	180	180	180	180	180	0
	13	167	167	166	166	166	176	176	176	176	176	176	181	181	-99	181	181	181	181	0
	14	167	167	166	166	166	176	176	176	176	176	176	182	182	182	-99	182	182	182	0
15	167	167	166	166	166	176	176	176	176	176	176	183	183	183	183	-99	183	183	0	
S w i t c h e s	16	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99
	17	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99
	18	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99

Figure C-1 16-Workstations, 3-Switches Routing Table

### C.2.2 16-Workstations, 4-Switches Topology Number 1

Figure C-2 displays the route table used for the first "16-4" topology tested. To obtain a route, first a node will look in its column's entry for the destination's corresponding row. The entry will give a 3-digit number, unless the node is switch, where the first two digits correspond to the next column to access. The last digit gives the output port for the next switch the worm will encounter. For the table in Figure C-2, the route flits' data fields corresponding to a source of 0 and a destination of 10 are 6, 6, 2, and 10. The latter is the final route flit. The actual corresponding table entries were as follows: 166, 196, 182, and 0. The zero signifies the fact that the last entry is the actual destination node. The simulation will then enter 10 into the route flit's data field to allow the destination node to check for an accurate routing.

		Sending Node															Switches					
R e c e i v i n g  N o d e		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
		-99	163	163	163	174	174	175	175	187	187	187	187	187	195	195	194	194	0	163	194	163
		162	-99	162	162	174	174	175	175	186	186	186	186	186	195	195	194	194	0	162	195	162
		161	161	-99	161	174	174	175	175	185	185	185	185	185	195	195	194	194	0	161	174	161
		160	160	160	-99	174	174	175	175	184	184	184	184	184	195	195	194	194	0	160	175	160
		164	164	164	164	-99	170	170	170	184	184	185	185	185	194	194	194	194	170	0	170	164
		164	164	164	164	171	-99	171	171	184	184	185	185	185	195	195	195	195	171	0	171	165
		165	165	165	165	172	172	-99	172	184	184	185	185	185	196	196	196	196	172	0	172	185
		165	165	165	165	173	173	173	-99	184	184	185	185	185	196	196	196	196	173	0	173	184
		164	164	164	164	176	176	177	177	-99	180	180	180	180	197	197	196	196	177	180	0	180
		165	165	165	165	176	176	177	177	181	-99	181	181	181	197	197	196	196	176	181	0	181
		166	166	166	166	176	176	177	177	182	182	-99	182	197	197	197	196	196	196	182	0	182
N o d e		167	167	167	167	176	176	177	177	183	183	183	-99	197	197	196	196	196	197	183	0	183
		167	167	166	166	176	176	176	176	186	186	187	187	-99	193	193	193	193	187	193	0	
		167	167	166	166	177	177	177	177	186	186	187	187	192	-99	192	192	192	186	192	0	
		167	167	166	166	175	175	175	175	186	186	187	187	191	191	-99	191	191	166	191	0	
		167	167	166	166	174	174	174	174	186	186	187	187	190	190	190	-99	190	167	190	0	
	S w i t c h e s	16	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99
		17	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99
18		-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	
19		-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	

Figure C-2 16-Workstations, 4-Switch Route Table for Configuration Number 1

### C.2.3 16-Workstations, 4-Switches Topology Number 2

Figure C-3 displays the route table used for the second “16-4” topology tested. To obtain a route, first a node will look in its column's entry for the destination's corresponding row. The entry will give a 3-digit number, unless the node is switch, where the first two digits correspond to the next column to access. The last digit gives the output port for the next switch the worm will encounter. For the table in Figure C-3, the route flits' data fields corresponding to a source of 0 and a destination of 10 are 6, 2, and 10. The latter is the final route flit. The actual corresponding table entries were as follows: 166, 182, and 0. The zero signifies the fact that the last entry is the actual destination node. The simulation will then enter 10 into the route flit's data field to allow the destination node to check for an accurate routing.

		Sending Node															Switches				
R e c e i v i n g  N o d e		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	0	-99	163	163	163	174	174	175	175	185	185	185	185	194	194	194	194	0	163	163	163
	1	162	-99	162	162	174	174	175	175	185	185	185	185	194	194	194	194	0	162	162	162
	2	161	161	-99	161	174	174	175	175	185	185	185	185	194	194	194	194	0	161	161	161
	3	160	160	160	-99	174	174	175	175	185	185	185	185	194	194	194	194	0	160	160	160
	4	164	164	164	164	-99	170	170	170	184	184	184	184	195	195	195	195	170	0	170	170
	5	164	164	164	164	171	-99	171	171	184	184	184	184	195	195	195	195	171	0	171	171
	6	165	165	165	165	172	172	-99	172	184	184	184	184	195	195	195	195	172	0	172	172
	7	165	165	165	165	173	173	173	-99	184	184	184	184	195	195	195	195	173	0	173	173
	8	166	166	166	166	177	177	177	177	-99	180	180	180	197	197	196	196	180	180	0	180
	9	166	166	166	166	177	177	177	177	181	-99	181	181	197	197	196	196	181	181	0	181
	10	166	166	166	166	177	177	177	177	182	182	-99	182	197	197	196	196	182	182	0	182
	11	166	166	166	166	177	177	177	177	183	183	183	-99	197	197	196	196	183	183	0	183
	12	167	167	167	167	176	176	176	176	186	186	187	187	-99	193	193	193	193	193	193	0
	13	167	167	167	167	176	176	176	176	186	186	187	187	192	-99	192	192	192	192	192	0
	14	167	167	167	167	176	176	176	176	186	186	187	187	191	191	-99	191	191	191	191	0
15	167	167	167	167	176	176	176	176	186	186	187	187	190	190	190	-99	190	190	190	0	
S w i t c h e s	16	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	
	17	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	
	18	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	
	19	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	

Figure C-3 16-Workstations, 4-Switches Route Table for Configuration Number 2

## Appendix D: Bibliography

- [AnC94] Anderson, Thomas E., Culler, David E., Patterson, David A., and the NOW team, A Case for NOW (Networks of Workstations), *IEEE Micro*, Feb. 95, vol.15, no. 1, p. 54-64.
- [BaC96] Banks, Jerry, Carson, John S. II, and Nelson, Barry L., Discrete-Event System Simulation, 2nd ed., Prentice-Hall, Inc., 1996.
- [BoC95] Boden, Nanette J., Cohen, Danny, Felderman, Robert E., Kulawik, Alan E., Seitz, Charles L., Seizovic, Jakov N., and Su, Wen-King, Myrinet -- A Gigabit-per-Second Local-Area Network, *IEEE Micro*, Feb 95 vol. 15, no. 1, p. 29-36.
- [CIJ89] Clark, David, Jacobson, Van, Romkey, John, and Salwen, Howard, An Analysis of TCP Processing Overhead, *IEEE Communication Magazine*, 27(6):23-29, Jun 1989.
- [FeD94] Felderman, Robert, DeSchon, Annette, Cohen, Danny, and Finn, Gregory, ATOMIC: A High Speed Local Communication Architecture, *Journal of High Speed Networks*, Vol. 3, No. 1 (94), pp. 1-29.
- [GeP96] Gerla, Mario, Palnati, Prasasth, and Walton, Simon, Multicasting Protocols for High-Speed Wormhole-Routing Local Area Networks, *Computer Communication Review*, Vol. 26, No. 4, pp. 184-93.
- [Gus90] Gusella, Riccardo, A Measurement of Diskless Workstation Traffic on Ethernet, *IEEE Transactions on Communications*, 38(9), Sep 90.
- [Hu96] Hu, Po-Chi, High-Speed Local Area Networks Using Wormhole Routing: Modeling and Extensions, Ph. D. Dissertation, University of California at Los Angeles, 1996.
- [Jai91] Jain, Raj, The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling, John Wiley & Sons, Inc., 1991.
- [LaC97] Lauria, Mario and Chien, Andrew, MPI-FM: High Performance MPI on Workstation Clusters, *Journal of Parallel and Distributed Computing*, Vol. 40, No. 1 (97), pp. 4-18.
- [LeT94] Leland, Will E., Taqqu, Murad S., Willinger, Walter, and Wilson, Daniel V., ON the Self-similar Nature of Ethernet Traffic (Extended Version), *IEEE/ACM Transactions on Networking*, 2(1), Feb 94, pp. 1-15.
- [LiM94] Liu, Lok Tin, Mainwaring, Alan, and Yoshikawa, Chad, White Paper on Building TCP/IP Active Messages, Nov 94.



- [MaS97] Mainwaring, Alan M. and Schleimer, Saul, System Area Mapping, To appear in the *Proceedings of the 9th Annual Symposium on Parallel Algorithms and Architectures (SPAA'97)*, 1997.
- [PaC95] Pakin, Scott, Lauria, Mario, and Chien, Andrew A., High Performance Messaging on Workstations: Illinois Fast Messages (FM) for Myrinet, *Proceedings of the High Performance Computing and Communications Supercomputing '95 Conference*, IEEE 1995, pp. 1528-1557.
- [PaF95] Paxson, Vern and Floyd, Sally, Wide-Area Traffic: The Failure of Poisson Modeling, *IEEE/ACM Transactions on Networking*, Vol. 3(3), Jun 95, pp. 226-244.
- [PaK97] Pakin, Scott, Karamcheti, Vijay, and Chien, Andrew A., Fast Messages (FM): Efficient, Portable Communication for Workstation Clusters and Massively-Parallel Processors, *IEEE Concurrency*, Vol. 5, no.2, p. 60-72.
- [Pax95] Paxson, Vern, Fast Approximation of Self-Similar Network Traffic, Technical Report Number LBL-36750, Lawrence Berkeley Laboratory and the University of California, Berkeley, Apr 95.
- [SeB93] Seitz, Charles L., Boden, Nanette J., Seizovic, Jakov N., and Su, Wen-King, The Design of the Caltech Mosaic C Multicomputer, *Proceedings of the University of Washington Symposium on Integrated Systems*, MIT Press, 1995, pp. 1-22.
- [TeH96] Tezuka, Hiroshi, Hori, Atsushi, and Ishikawa, Yutaka, PM: A High-Performance Communication Library for Multi-User Parallel Environments, Technical Report Number TR-96-015 Tsukuba Research Center, Real World Computing Partnership, Nov 96.
- [VeL96] Verstoep, Kees, Langendoen, Koen, and Bal, Henri, Efficient Reliable Multicast on Myrinet, *Proceedings of 25th International Conference on Parallel Processing 12-16 Aug 96*, IEEE, Vol 3, pp. 156-165.
- [voC92] von Eicken, Thorsten, Culler, David E., Goldstein, Seth Copen, and Schausser, Klaus Erik, Active Messages: a Mechanism for Integrated Communication and Computation, UC Berkeley Report No. UCB/CSD 92/#675, Mar 92.
- [WaM96] Wang, Lei and McCrosky, Carl, Self-similarity of Aggregated Network Traffic Models, IFIP Workshop TC6. *Proceedings of the Third Workshop on Performance Modeling and Evaluation of ATM Networks*, 2-6 Jul 95., IEEE 1996, p. 860, 38/1-13.
- [ZhY95] Zhang, Xiaodong and Yan, Yong, Modeling and Characterizing Parallel Computing Performance on Heterogeneous Networks of Workstations, *Proceedings of the 7th IEEE Symposium on Parallel and Distributed Processing*, IEEE Computer Society Press, Oct 95.

## **Appendix E: Vita**

First Lieutenant Dustin E. Yates was born August 22, 1970 in Waco, Texas. After graduating from Ardmore High School, Ardmore, Oklahoma in 1988, he enlisted in the United States Army where he received the Distinguished Graduate Award for his Computer Programmer training. He completed his Bachelor of Science Degree in Computer Science at the University of North Texas in May 1994. Upon graduation, he was commissioned through the Air Force ROTC program and was subsequently assigned to the 333d Training Squadron at Keesler, AFB, Biloxi, Mississippi in September 1994. While at Keesler, Lieutenant Yates served as a computer networks instructor for the Basic Communications-Computer Officer Training Course and as a GCCS Systems instructor until May 1996. Upon transfer in May of 1996, he entered the Computer Engineering Program at the School of Engineering, Air Force Institute of Technology.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE December 1997		3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE Modeling and Simulation Support for Parallel Algorithms in a High-Speed Network			5. FUNDING NUMBERS	
6. AUTHOR(S) Dustin E. Yates, First Lieutenant, USAF				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology 2750 P. St. WPAFB OH 45433-7126			8. PERFORMING ORGANIZATION REPORT NUMBER  AFIT/GCS/ENG/97D-20	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Rome Labs, RL/OCSS Surveillance/Signal Processing Branch 26 Electronic Parkway Rome, NY 13441-4514			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for public release; distribution unlimited			12b. DISTRIBUTION CODE  A	
13. ABSTRACT (Maximum 200 words) This thesis investigates the ability of a simulation model to compare and contrast parallel processing algorithms in a high-speed network. The model extends existing modeling, analysis, and comparison of parallel algorithms by providing graphics based components that facilitate the measurement of system resources. Simulation components are based on the Myrinet local area network standard. The models provide seven different topologies to contrast the performance of five variations of Fast Fourier Transform (FFT) algorithms. Furthermore, the models were implemented using a commercially developed product that facilitates the testing of additional topologies and the investigation of hardware variations. Accurate comparisons are statistically validated and supported via common operating assumptions and the Myrinet standards. Based on the statistical confidence, the conclusion is drawn that a variation of a FFT algorithm based on row-column computations performs better than the other choices considered.				
14. SUBJECT TERMS Modeling, Simulation, Myrinet, Parallel Processing, Network, LAN, Local Area Network, High-Speed Network			15. NUMBER OF PAGES 159	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	